Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Znalostný manažment na báze technológie .NET

Projektová dokumentácia

Tímový projekt

Autori: Bc. Vladimír Mlynarovič
Bc. Michal Pažitný
Bc. Zdenko Porubčan
Bc. Daniel Princzkel
Bc. Lukáš Sim

Tím: FOOBE (tím č.19)
Vedúci tímu: Ing. Ivan Polášek, PhD.

Dátum: 20. 5. 2008

Obsah

0 Úvod

0.1 Úvod do problematiky

Znalostný manažment a znalostná ekonomika pútajú čím ďalej, tým väčšiu pozornosť modernej spoločnosti. Na základe toho sa objavujú zaujímavé projekty zamerané na túto tému, ako systémy znalostného riadenia podnikov, či celých spoločností. Podobná búrlivá diskusia sa vedie aj ohľadom sémantického webu. Aj napriek veľkej snahe v oblasti výskumu, stále chýbajú zdroje a prostriedky, pričom používateľsky vhodný a akceptovateľný systém neustále absentuje. Naďalej preto vzniká priestor pre vyvíjanie systémov, ktoré by boli schopné znalosti reprezentovať, spracovávať, vytvárať relácie s ostatnými znalosťami, modifikovať ich a pod.

0.2 Účel dokumentu

Snahou tohto projektu je preto vytvoriť dynamický sémantický model, ktorý by napomohol pri tvorbe nových dokumentov (zdrojových kódov, dokumentácií a manuálov a pod.) pomocou získaných znalostí. Touto témou sa už v predošlom roku zaoberal tím *Lucky Number 7*, ktorého výsledky by mali priniesť značnú pomoc a umožniť preniesť sa rýchlejšie cez už prekonané problémy. Úlohou nášho tímu je zároveň posunúť sa v tejto oblasti ďalej, teda predostrieť nové možnosti, myšlienky či nápady, o ktorý bude v konečnej fáze samotný systém obohatený.

0.3 Prehľad dokumentu

Celý dokument je členený do niekoľkých základných častí a troch príloh.

Prvou je analýza, ktorá je venovaná znalostiam a znalostnému manažmentu. Po nej nasleduje analýza znalostného systému vytvoreného tímom *Lucky Number 7* minulý rok. Zvyšnú časť analýzy tvoria podrobnejšie úvahy o možných riešeniach a koncepciách v projekte.

Špecifikáciu požiadaviek tvoria najmä dva modely. Prvým je model prípadov použitia, ktorého dopĺňa popis identifikovaných hráčov v systéme a tabuľky jednotlivých prípadov použitia. Druhým modelom je dátový model, konkrétne jeho logická úroveň. Kapitolu špecifikácie uzatvárajú nefunkcionálne požiadavky na systém.

Tretia kapitola má názov návrh. V jej úvode je naznačený pohľad na celkovú architektúru systému a fyzický model údajov. Architektúra delí systém do jednotlivých modulov, ktorých činnosť je neskôr vysvetlená podrobnejšie.

Štvrtá kapitola tvorí technickú dokumentáciu k produktu a stručný opis jeho implementácie. Sú tu bližšie opísané najdôležitejšie časti implementácie jednotivých modulov vrátane diagramov tried.

Nasleduje kapitola pojednávajúca o testovaní produktu spolu s testovacími scenármi.

Záverečná časť hodnotí projekt, splnenie/nesplnenie požiadaviek kladených v špecifikácii, predkladá možné vylepšenia a rozšírenia sytému.

V príloje A je Používateľská príručka, v prílohe B Inštalačná príručka a v prílohe C sa nachádza Revízia tejto dokumentácie.

0.4 Použité skratky

DPL – (Data Provider Layer) je vrstva poskytovania dát vyvinutý a štandardizovaný konzorciom W3C (World Wide Web Consortium)

OWL – (Web Ontology Language) jazyk na definovanie webových ontológií

RDF – (Resource Description Framework) je World Wide Web Consortium (W3C) špecifikácia formátu podporujúca popis zdrojov pomocou abstraktného modelu metadát, implementovaná v jazyku XML

UC – (Use Case) prípad použitia

VD – výber dokumentov

XML – (eXtensible Markup Language) v preklade rozšíriteľný značkovací jazyk, ktorý bol vyvinutý a štandardizovaný konzorciom W3C (World Wide Web Consortium)

1 Analýza

1.1 Úvod do manažmentu znalostí

Informačná spoločnosť, ako sa súčasná spoločnosť zvykne označovať, čelí nespočetnému množstvu informácií z veľkého množstva zdrojov. Prudký nárast množstva informácií za posledné obdobie so sebou však priniesol i nežiaduce problémy. Ťažkosti s prístupom, extrahovaním, interpretáciou, udržovaním informácií zostávajú na používateľovi. Myšlienka reprezentácie informácií vo forme znalostí nie je najmladšia, napriek tomu však táto téma ostáva stále otvorená a prebádaná iba čiastočne. Práve znalosti by mali posunúť informácie na vyššiu úroveň v reprezentácií, prehľadnosti či prístupe a vyhľadávaní.

Zavádzania manažmentu znalostí do praxe podnecujú hlavne tieto tri faktory [4]:

1. Od času potrebného na nasadenie výrobku na trh je závislý úspech či neúspech daného produktu. Pre spoločnosti je nevyhnutné využiť výhody prvotného iniciátora na trhu alebo aspoň rýchle vytvorenie konkurencie. Z tohto dôvodu je pre firmy nevyhnutné plne využitie ich znalostí týkajúcich sa trhu, zákazníkov, konkurenčných firiem a dostupných technológií.

2. Iná situácia vzniká pri fúzii alebo reštrukturalizácii firiem, ktoré hlboko závisia na nadobudnutých znalostiach. Pri týchto operáciách je veľmi dôležité vedieť preniesť znalosti z jednej organizácie do druhej a určiť, ktoré zo znalostí sú užitočné pre čo najrýchlejší profit firmy a čo najdlhšie prežitie na trhu.

3. Moderné informačné technológie poskytujú obrovské množstvo informácií a pomerne jednoduchý prístup k nim. Týchto informácií je veľké množstvo a je priveľmi zložité nájsť požadovanú informáciu dostatočne rýchlo na to, aby sme ju mohli využiť a aby námaha vynaložená na jej získanie neprekročila užitočnú hodnotu tejto informácie.

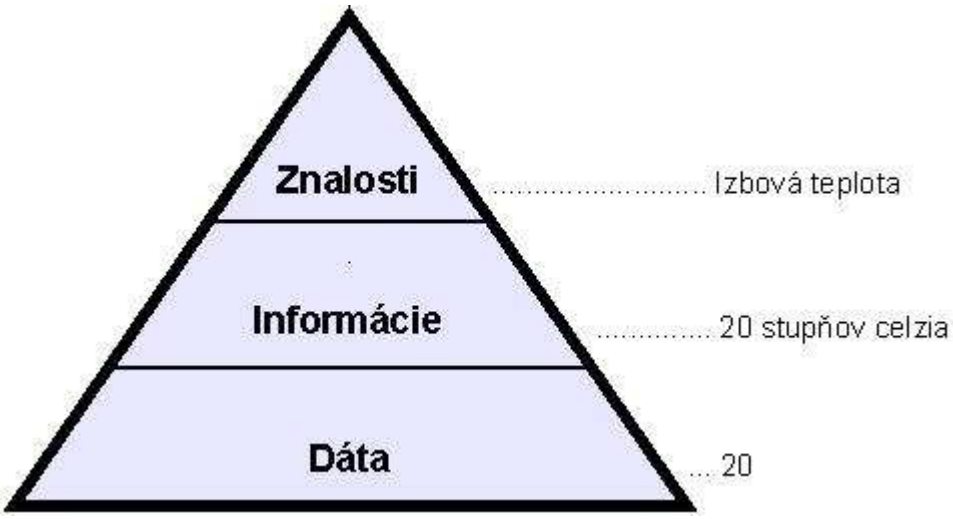
Manažment znalostí čerpá zo širokej škály disciplín ako napríklad [8]:

- Umelá inteligencia, expertné a znalostné systémy
- Počítačom podporovaná spolupráca (groupware)
- Informatika, kognitívne vedy, filozofia
- Správa dokumentov
- Systémy pre podporu rozhodovania
- Re-inžiniering firemných procesov
- Riadenie ľudských zdrojov
- Organizačná kultúra a pod.

Základné pojmy

Manažment znalostí
„Systémy pre vyhľadávanie a spracovanie znalostí a všeobecne prácu so znalosťami označujeme ako manažment znalostí. Aby systém vedel so znalosťami správne zaobchádzať, musí ich vedieť kategorizovať, analyzovať, dalo by sa povedať, že rozumieť im“.[3]

Na obr. 1 je zobrazená hierarchia pojmov v znalostnom manažmente. Pri každom pojme je uvedený aj príklad.



Obr. 1. Hierarchia pojmov v znalostnom manažmente

Dáta
Dáta sú súbor znakov predstavujúcich diskrétne objektívne fakty. Ako také majú dáta veľmi malú výpovednú hodnotu, ale dajú sa jednoducho uchovávať a spracovávať vo výpočtových systémoch.

Informácie
Informácie sú dáta ktoré sú vložené do kontextu. Informácie majú väčšiu hodnotu ako dáta no môžu spôsobovať dvojznačnosť alebo stratiť hodnotu, ak príjemca informácie nerozumie kontextu.

Znalosti
Znalosti sú informácie vložené do konkrétneho kontextu a sú hodnotné pre toho kto ich pozná. Umožňujú mu vykonávať činnosti, ktoré by bez týchto znalostí vykonávať nemohol.

1.2 Analýza znalostného systému vytvoreného tímom Lucky Number 7

Tím Lucky 7 sa zaoberal návrhom a implementáciou produktu, ktorý je založený na viacvrstvovej architektúre, kde každá vrstva má presne definovanú vlastnú funkcionalitu. Z projektovej dokumentácie však nie je veľmi jasné, z koľkých vrstiev táto architektúra pozostáva. Na lepšie porozumenie tohto návrhu je vhodné doplniť jednoduchý diagram, ktorý by znázornil jednotlivé úrovne a ich poradie. Vykonali sme analýzu jednotlivých vrstiev.

Databázová vrstva

Databázová vrstva slúži na ukladanie a získavanie dokumentov z externej databázy (MS SQL Server 2005 Express Edition). Tvorí rozhranie medzi vyššou vrstvou (Data Provider Layer) a touto databázou. Návrh databázovej vrstvy je rozdelený do dvoch častí.

Prvá časť opisuje návrh dátovej štruktúry, ktorá definuje uchovávaný dokument v databáze. Ide konkrétne o tieto údajové položky:

- identifikátor súboru
- meno súboru (meno súboru zadané pri ukladaní do úložišta)
- obsah súboru
- metadáta opisujúce súbor

- hash jednoznačne identifikujúci súbor

Druhá časť návrhu sa zameriava na opis služieb:

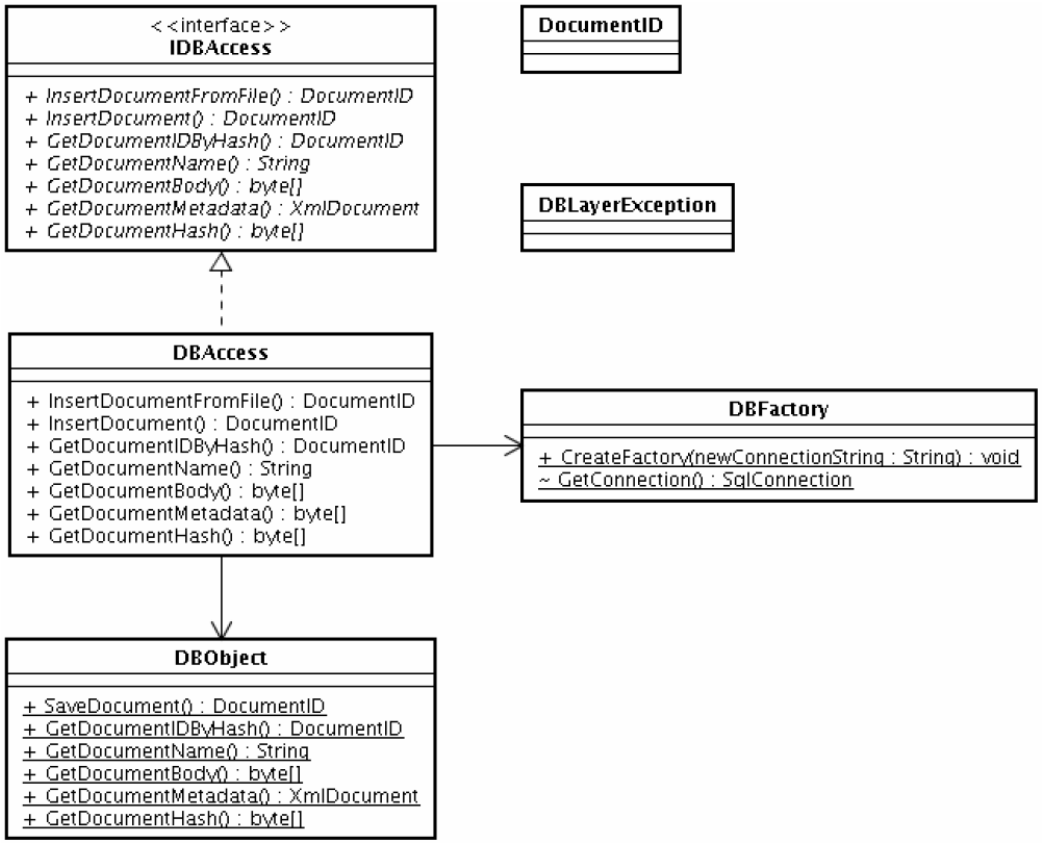
- pridanie dokumentu
- získanie ID dokumentu
- zistenie mena dokumentu
- získanie obsahu dokumentu
- získanie metadát
- získanie hashu

a s nimi súvisiacimi funkcií rozhrania *IDBAccess*:

- `DocumentID InsertDocumentFromFile(String name, String file, XmlDocument metadata, byte[] hash);`
- `String GetDocumentIDByHash(byte[] hash);`
- `String GetDocumentName(DocumentID ID);`
- `byte[] GetDocumentBody(DocumentID ID);`
- `byte[] GetDocumentMetadata(DocumentID ID);`
- `byte[] GetDocumentHash(DocumentID ID);`

ktoré poskytuje databázová vrstva. Tieto funkcie sú vysvetlené podrobne, a teda vieme presne povedať, čo ktorá funkcia vykonáva. Sú definované hlavičky metód, výnimky, vstupy a výstupy.

Pri práci so súbormi sa využíva trieda DocumentID, ktorá zapuzdruje identifikáciu dokumentu v rámci vrstvy. Chybové stavy vznikajúce na tejto vrstve sú obalené do výnimky *DBLayerException*. K pochopeniu celého návrhu a implementácie poslúži nasledovný diagram tried (obr. 2) [1].



Obr. 2. Diagram tried databázovej vrstvy

Vrstva poskytovania dát

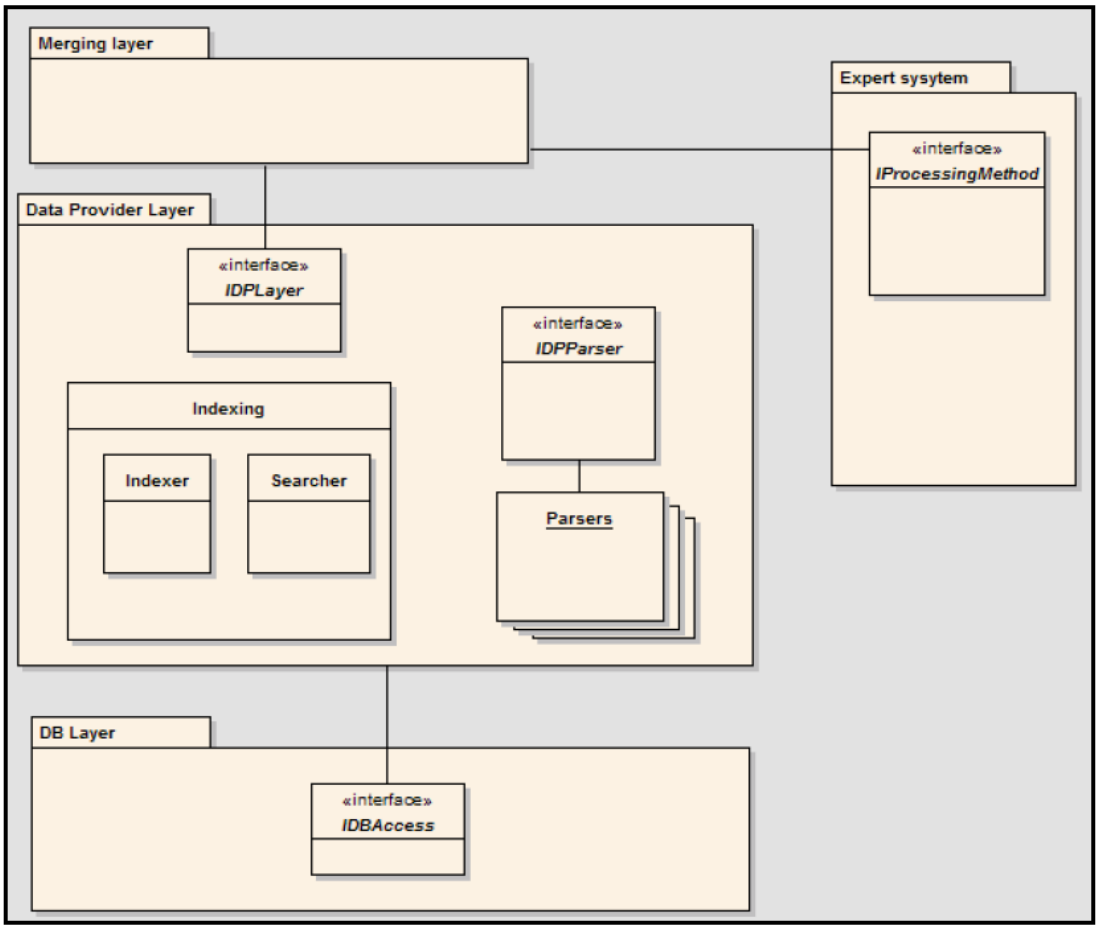
Ďalšou vrstvou, ktorú tím Lucky7 navrhol, je vrstva poskytovania dát (Data provider layer, DPL). Takisto ako pri databázovej vrstve autori sa venovali návrhu a implementácii tejto vrstvy, ako aj opisu vrstvy a jednotlivých požiadaviek.

Úlohou DPL vrstvy je spracovanie údajov zo zadaných dokumentov, ich indexácia a vyhľadávanie. Boli identifikované dva hlavné moduly (parsovací a indexovací modul), ktorých funkciu opísali stručne, ale zato výstižne.

Ako parsovací modul bol vytvorený komponent, ktorý dokáže analyzovať súbory typu .doc a .docx. Bol vytvorený na základe volania metód menného priestoru Microsoft.Office.Interop a využíva pre svoju činnosť aplikáciu MS Word 2007. Preto je pre používanie systému potrebné, aby bola táto aplikácia dostupná na počítači tvoriacom server aplikácie.

Druhý modul slúži pre indexáciu dokumentov podľa ich kapitol v slovnom indexe a následné vyhľadávanie v tomto indexe na základe kľúčových slov. Na vytvorenie tejto funkcionality bola použitá dynamická knižnica pre indexovanie textu s názvom Lucene.Net.

Pre lepšie pochopenie štruktúry týchto modulov v rámci vrstvy, ale aj vrstvy v rámci celej architektúry systému, slúži nasledovný diagram komponentov (obr. 3) [1].



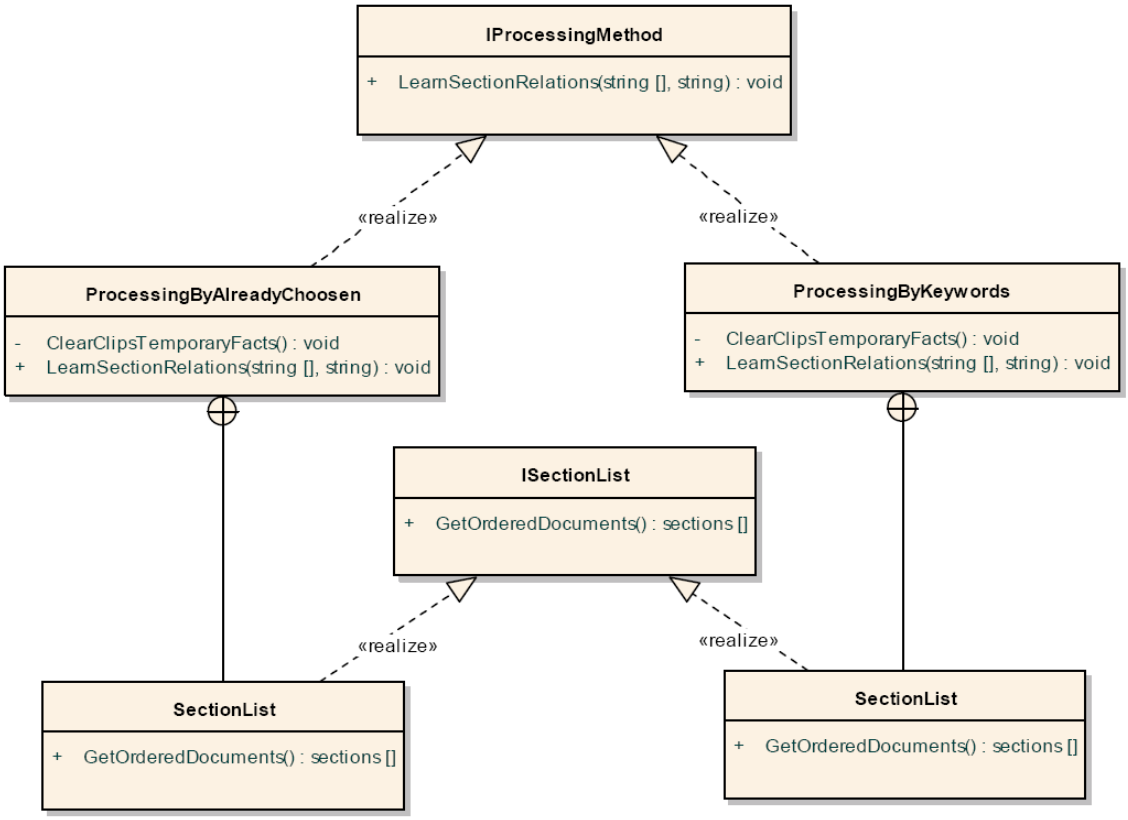
Obr. 3. Štruktúra vrstvy DPL

Komponent - expertný systém

Pre účely expertného systému si tím Lucky7 vybral systém Clips. Venovali sa implementácii jeho integrácie do systému, ako aj metódam vyhľadávania dokumentov. Z definovanej špecifikácie expertného systému sme sa dozvedeli úlohu komponentu v systéme:

- inteligentné vyhľadávanie na základe kľúčových slov
- inteligentné vyhľadávanie na základe kapitol už nachádzajúcich sa v dokumente
- učenie sa na používateľovom výbere kapitoly podľa kľúčových slov
- učenie sa na používateľovom výbere kapitoly na základe kapitol už nachádzajúcich sav dokumente

Návrh modulu je rozdelený na všeobecnú časť nezávislú od použitého expertného systému a časť implementujúcu techniky prehľadávania a dopĺňania znalostí v Clipse. Každá časť je opísaná samostatne na základe svojich tried. Diagram tried uľahčuje pochopenie architektúry daných častí (obr. 4) [1].



Obr. 4. Architektúra modulu expertný systém

Súčasťou návrhu je aj opis jednotlivých techník používaných na prehľadávanie dokumentov – technika prehľadávania na základe kľúčových slov a technika prehľadávania na základe už existujúcich kapitol. Každá z týchto techník pracuje s rozličnými pojmami a dátovými štruktúrami.

Plugin do MS Word 2007

Predstavuje prístup k integrácii projektu do programu MS Word. Plugin bol vyvinutý v prostredí .NET Framework. V rámci tohto prostredia vytvorili autori 4 projekty pri vývoji plugin-u. Každý z nich zohráva

určitú úlohu pri vývoji a testovaní plugin-u – webové služby (napr. na testovanie), vytváranie inštalátora, nastavenie bezpečnosti a samotný projekt s plugin-om – GUIKrtko. O projekte GUIKrtko autori informujú trochu viac ako o podporných projektoch, ale aj tak sa nedozvedáme informácie o návrhu a bližšom fungovaní samotného plugin-u. Samotnú funkcionalitu sa nám nepodarilo overiť.

1.3 Možnosti spracovania dokumentov

Táto časť projektu sa zaoberá analýzou súčasného stavu v oblasti získavania a vyhľadávania dokumentov. Pred samotným spracovaním dokumentov je nutné definovať, ako majú byť jednotlivé dokumenty reprezentované. Analyzovali sme preto uvedené často používané spôsoby reprezentácie jednotlivých dokumentov, ktoré sa využívajú v tejto oblasti a pokúsili sa o ich porovnanie.

Výber dokumentov

Výber dokumentov (VD) je výpočtový proces selekcie podmnožiny dokumentov z celkovej databázy dokumentov, ktorého výsledok je v sumárnej forme zobrazovaný používateľovi, zvyčajne ako odozva na užıvateľovu požiadavku. VD systém teda rozdeľuje dokumenty do dvoch tried:

- dokumenty relevantné k používateľovej požiadavke, ktorých zoznam sa používateľovi zobrazuje v prijateľnej forme ako výstup VD procesu.

- dokumenty nespĺňajúce požiadavku používateľa, ktoré nie sú používateľovi zobrazované.

Našou úlohou nie je len navrhnúť VD systém vykonávajúci selekciu dokumentov, ale taktiež zotriedenie podľa stupňa relevancie k požiadavke. Vypočítavať teda stupeň príslušnosti jednotlivých dokumentov databázy k príslušným triedam. Vo všeobecnosti VD proces môžeme opísať 4 základnými fázami:

I. **indexovanie** – množina vstupných dokumentov je pri určitej textovej reprezentácii konvertovaná do výrazov tejto reprezentácie. Tieto výrazy nazývame *reprezentanti dokumentov*, pričom ich štruktúra musí byť použiteľná pre ďalšie fázy VD procesu.

II. **formulácia požiadavky** - používateľ musí zadať čiastočnú informáciu o relevantnosti dokumentov vo forme otázky (request), ktorá je interpretovateľná VD systémom. Otázka sa často zadáva vo forme zrozumiteľnej pre systém, pričom často obsahuje boolovské operátory (and, or, not), prípadne iné (proximity operátor - near). Ďalším spôsobom zadania otázky je zadanie priamou formou v prirodzenom jazyku, pri ktorom si systém vyextrahuje dôležité slová, frázy pre príznaky charakterizujúce požiadavku.

III. **porovnanie –** systém musí implicitne alebo explicitne porovnávať používateľskú otázku s reprezentantmi dokumentov a vytvoriť tak klasifikačné rozhodnutie o tom, ktoré dokumenty vyberieme, t.j. vyhovujú danej požiadavke, a v akom poradí. Vybrané dokumenty sa potom zobrazia používateľovi.

IV. **spätná väzba –** málokedy sa stáva, že prvotný, počiatočný výber dokumentov odpovedá potrebám používateľa. Zvyčajne je potreba niekoľkých iterácií, v ktorých sa požiadavka (otázka) používateľa modifikuje zadaním novej alebo úpravou starej otázky. Táto fáza sa zvyčajne označuje ako *relevance feedback*.

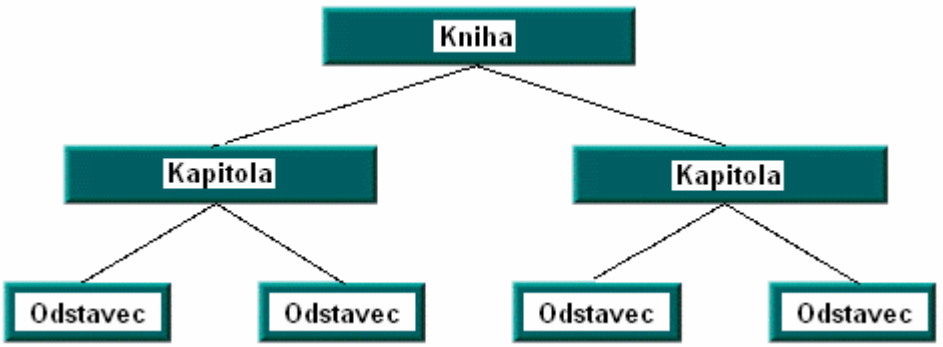
Reprezentácia textových dokumentov

Jednotlivé operácie pre spracovanie dokumentov, ku ktorým sa radí aj klasifikácia prebiehajú nad vstupným príkladovým priestorom tzv. *korpusom*. Korpusom teda nazývame vstupnú kolekciu dokumentov určenú na ďalšie spracovanie. Každý dokument je tvorený kolekciou termov z *univerza* termov. Pod pojmom *term*

budeme rozumieť vybranú textuálnu jednotku z textu. Môže ňou byť napr. slovo, kmeň slova (stem), fráza, lema, a pod.

Z hľadiska štruktúry sa na každý dokument môžeme pozerať ako na:

- jednoduché textové pole, pri ktorom nezohľadňujeme štruktúru daného dokumentu
- štruktúrovaný celok tvorený na základe určitých pravidiel pre štruktúrovanie dokumentu, podľa ktorých je dokument rozčlenený do kapitol, odstavcov, a pod. Ukážka štruktúrovaného dokumentu je znázornená na nasledujúcom obrázku (obr. 5)



Obr. 5. Zobrazenie štruktúry dokumentu

Ľubovoľný textový dokument nie je len sústavou slov, fráz a viet v zmysle syntaktickej stavby daného textu, ale je tvorený taktiež sémantickými jednotkami popisujúcimi význam jednotlivých termov. Porozumenie textu si vyžaduje lingvistické znalosti týkajúce sa morfológie slov, sémantiky (významu) slov, štruktúry viet, susednosti slov, a pod. Tieto dodatočné informácie sa môžu zahrnuť do reprezentácie dokumentu, pre lepšie pochopenie jeho obsahu.

Medzi základné problémy, s ktorými sa stretávame pri tvorbe reprezentácie dokumentov, patria:

- **polysemický problém -** jedno slovo má viacero významov
- **synonymický problém -** ten istý pojem, resp. jeho význam sa dá vyjadriť viacerými slovami

Voľbou vhodnej reprezentácie dokumentov dokážeme tieto problémy vo väčšej, či menšej miere potlačiť. V nasledujúcich kapitolách sú uvedené najčastejšie spôsoby reprezentácie dokumentov.

Teoretické modely

Booleovský model reprezentuje dokumenty pomocou skupiny slov, pričom každé slovo predstavuje Booleovskú premennú. Jej hodnota je *Pravda*, ak sa slovo v dokumente nachádza. Váženie (váhovanie) slov, t. j. nakoľko dané slovo reprezentuje obsah dokumentu, nie je povolené. Pri vyhľadávaní dokumentov sa dopyty reprezentujú ako ľubovoľné Booleovské výrazy spojené štandardnými logickými operátormi: AND, OR a NOT.

Fuzzy model sa zakladá na teórii fuzzy množín, ktorá povoľuje čiastočné členstvo prvku v množine - v porovnaní s klasickou teóriou množín, v ktorej prvok buď patrí do množiny alebo nepatrí. Logické operátory sú náležite preddefinované tak, aby zahŕňali čiastočnú príslušnosť k množine. IR modely založené na teórii fuzzy množín sa ukázali ako neschopné rozlišovať relevantné dokumenty medzi vyhľadávanými [6].

Prísne Booleovské a fuzzy modely sú výhodnejšie ako iné modely v zmysle výpočtových nárokov. Malá pamäťová náročnosť na reprezentáciu dokumentu a malá algoritmická zložitosť indexovania a výpočtu podobnosti dopyt – dokument.

Vektorový model

Vector Space (VS) model je najčastejšou a zároveň najjednoduchšou reprezentáciou textových dokumentov. Príznakový priestor pre túto reprezentáciu je konštruovaný na základe množiny slov, pričom každý príznak korešponduje s textovou jednotkou v korpuse. Za textovú jednotku je považované slovo. To znamená, že dokument sa tu chápe ako sekvencia slov bez akýchkoľvek iných znalostí o štruktúre textu, pričom sa neberie ohľad na sémanticky význam jednotlivých slov.

Tento model vychádza len zo štatistických charakteristík dokumentu, a to najmä z distribúcie pravdepodobnosti jednotlivých slov extrahovaných z korpusu a dopytov. Každý dokument je vyjadrený ako vektor d_i v n-rozmernom priestore R^n, kde každá os daného priestoru predstavuje jedno slovo.

Môžeme teda písať pre dokument dok_i :

$$d_i = (w_{i1}, w_{i2}, .., w_{ij}, ..., w_{in}) \quad (1)$$

kde w_{ij} – funkčná hodnota vyjadrujúca váhu, resp. dôležitosť *j*-tého slova v *i*- tom dokumente dok_i.

n – rozmer indexovej množiny slov, t.j. tých slov, ktoré tvoria príznaky popisu jednotlivých dokumentov.

Vektor d_i sa v literatúre označuje aj ako *lexikálny profil* dokumentu.

Celkový korpus *m* dokumentov môžeme tak vyjadriť maticou rozmeru *m x n*. Takáto matica sa častokrát v literatúre označuje ako *term-dokument* matica. Voľba jednotlivých slov opisujúcich korpus dokumentov by mala byť zvolená tak, aby dané slová mali čo najväčšiu rozlišovaciu schopnosť.

Potom je možné definovať operáciu podobnosti

$$sim(d_i,q) = (d_i * q) / (|d_i| \times |d_i|), \quad (2)$$

kde q je vektor reprezentujúci vyhľadávaciu požiadavku.

Možnosť, ako sa dá ovplyvniť reprezentácia dokumentu a tým aj výsledky vyhľadávania je ovplyvnenie váhy jednotlivých kľúčových slov rôznymi spôsobmi ich výpočtu:

- *TF-IDF váhová schéma*

  Wi,j = tfi,j x log (N/ni),

  kde

  N – celkový počet dokumentov

  ni – počet dokumentov, v ktorých sa ki vyskytuje

  fi – frekvencia výskytu ki

  tfi,j = fi,j / (maxi fi,j) je normalizovaná frekvencia pre dokument dj

- *Schéma podľa Saltona a Buckleyho*

Wi,j=(0,5 + (0,5 x fi,q)/max(Ifi,q)) x log(N/ni),

kde

Ifi = log (N/ni) je inverzná frekvencia dokumentu pre kľúčové slovo ki

- *Asociačnými pravidlami* [2]

    1. vygenerovanie množiny z kľúčových slov,
    2. vygenerovanie asociačných pravidiel spĺňajúcich minimálne kritériá.

Pravdepodobnostné modely

Pravdepodobnostný model uvažuje závislosti medzi slovami v dokumente a používa váhy slov zadaných v dopyte používateľa. Model je založený na dvoch hlavných parametroch – P_{rel} *a* P_{irel}, pravdepodobnosť relevantnosti a irelevantnosti dokumentu k dopytu, ktoré sa počítajú pomocou pravdepodobnostných váh slov a prítomnosti slov v dokumente. Pravdepodobnostný prístup vyžaduje trénovaciu množinu dokumentov, ktorú získa od používateľa tak, že používateľ poskytne rozhodnutie o relevantnosti dokumentu vzhľadom na výsledok dopytu.

Predpokladáme, že v množine máme N dokumentov, z ktorých je R relevantných pre používateľský dopyt, R_t relevantných dokumentov obsahuje slovo t a t sa vyskytuje v f_t dokumentoch. Potom môžeme pre každé slovo t vypočítať dve podmienené pravdepodobnosti:

P (t je v dokumente | dokument je relevantný) = R_t / R ;

P (t je v dokumente | dokument je irelevantný) = $(f_t - R_t) / (N - R)$.

Z týchto odhadov sa pomocou *Bayesovej teorémy* odvodí váha slova t ako:

$$w_t = \log \frac{R_t / (R - R_t)}{(f_t - R_t) / [N - f_t - (R - R_t)]} \tag{3}$$

Čitateľ (menovateľ) zlomku vyjadruje pravdepodobnosť, že slovo t sa nachádza v relevantnom (irelevantnom) dokumente. Váha $w_t > 0$ indikuje relevantnosť dokumentu k dopytu za podmienky, že sa slovo t vyskytuje v dokumente. Váha menšia ako nula indikuje jeho irelevantnosť.

Porovnanie modelov

Nasledujúca tabuľka (tab. 1) slúži pre porovnanie 3 klasických modelov, slúžiacich na reprezentáciu dokumentov.

Model	Výhody	Nevýhody
Boolovský	• jasný formalizmus • jednoduchosť	• presná zhoda výskytu termov otázky v dokumente môže viesť k príliš veľkému alebo naopak príliš malému počtu dokum. v odpovedi • dokumenty nemožno usporiadať podľa stupňa relevancie k otázke • neberie sa do úvahy frekvencia výskytu jednotlivých termov otázky v dokumente
Vektorový	• schéma váženia termov podľa frekvencie ich výskytu zvyšuje výkonnosť vyhľadávania • vyhľadá aj dokumenty, ktoré len čiastočne vyhovujú zadanej otázke usporiadania nájdených dokumentov podľa stupňa ich relevancie	• predpoklad nezávislosti indexových termov síce neplatí, ale prakticky ide väčšinou iba o lokálne závislosti malých skupín termov
Pravdepodobnostný	• usporiadanie nájdených dokumentov podľa pravdep. ich relevancie k otázke	• nutnosť počiatočného odhadu niektorých pravdepodobností • neberie sa do úvahy frekvencia výskytu jednotlivých termov otázky v dokumente • predpoklad nezávislosti indexových termov síce neplatí, ale prakticky ide väčšinou iba o lokálne závislosti malých skupín termov

Tab. 1. Porovnanie modelov

1.4 Vyhľadávanie informácií v dokumentoch, indexovanie

Keďže sekvenčné vyhľadávanie slov a fráz v texte je pri objemnej databáze veľmi neefektívne, vznikali metódy, ako hľadanie urýchliť. Zistilo sa, že je dobré vytvárať indexy slov, ktoré sa v dokumente vyskytujú. Indexovanie dokumentov značne urýchľuje vyhľadanie požadovaných slov v dokumente. Index neobsahuje všetky slová, ktoré sa v dokumente vyskytujú. Takzvané „stop slová“, ktoré sú v jazyku najčastejšie používané a/alebo nevytvárajú obraz o obsahu dokumentu sú z indexu vynechané. Patria tu napríklad slová *a, alebo, pre, od, do, iné*. Indexy majú zvyčajne veľkosť 20-30% z veľkosti dokumentu.

Inverzný index

Najbežnejšia metóda indexovania je tzv. invertované indexovanie [1]. Je to metóda, pri ktorej sa postupne vykonávajú tieto kroky:

- dokument sa rozdelí na slová
- vytvorí sa zoznam týchto slov
- ku každému slovu sa priradí zoznam pozícií jeho výskytov v dokumente

Pozície slov v takomto indexe môžu byť buď znakové, alebo slovné. Slovné pozície sú výhodnejšie pri vyhľadávaní frázy s približnou vzdialenosťou slov. Aby sme vedeli rozoznať, v ktorom dokumente a na akej pozícii sa hľadané slovo nachádza, musí mať slovo priradený jednoznačný identifikátor dokumentu a pozíciu slova. Pokiaľ index neobsahuje slovnú pozíciu, trvá vyhľadávanie frázy oveľa dlhšie.

Znakové pozície sú užitočné pri nachádzaní textu v dokumente. Inak je potrebné prehľadať celý dokument a nájsť hľadané slová. Je potrebné zvážiť, či potrebujeme obidva druhy pozícií a či sme ochotní zväčšiť tak výsledný indexový súbor.

Tretím parametrom môže byť číslo bloku. Dokumenty môžu byť delené na bloky s pevnou dĺžkou (10,64 alebo 256 tisíc znakov/slov). Použitím takýchto blokov sa veľkosť indexu zmenšuje.

Hľadanie v indexoch

Cieľom indexovania je teda zvýšenie rýchlosti vyhľadávania v dokumente. Vyhľadávanie môžeme rozdeliť na tri základné kroky:

1. Rozdelenie hľadanej požiadavky na slová a vyhľadanie každého slova v indexe.
2. Získanie zoznamu dokumentov, v ktorých sa slovo nachádza.
3. Skombinovanie zoznamov dokumentov podľa operátorov použitých v požiadavke.

V prvom kroku sa z požiadavky odstránia „stop slová“. Je možné, že ak sa požiadavka skladá iba z takýchto slov, po prvom kroku nemáme čo hľadať. V druhom kroku sa vytvoria zoznamy dokumentov prislúchajúce jednotlivým slovám. Podľa operátorov v požiadavke tieto zoznamy skombinujeme a dostaneme výsledný zoznam hľadaných dokumentov.

Príklad
Slová, ktoré hľadáme: *Jablko, Hruška*

Zoznamy dokumentov, ktoré obsahujú jednotlivé hľadané slová:
Jablko 21,68,125,348
Hruška 25,68,97,125,336,412

Výsledný zoznam dokumentov, ktoré obsahujú hľadané slová:
Pri použití operátora AND: 68,125
Pri použití operátora OR: 21,25,68,125,336,348,412

Správnosť a priepustnosť

Dve základné metriky pre vyhľadávanie slov v dokumente sú:

správnosť **-** pomer nájdených relevantných dokumentov ku všetkým nájdeným dokumentom

priepustnosť - pomer nájdených relevantných dokumentov ku všetkým relevantným dokumentom v báze dokumentov

Príklad:
Máme daný stav znázornený v tabuľke:

	Relevantné	Irelevantné	Spolu
Nájdené	40	80	120
Nenájdené	10	870	880
Spolu	50	950	1000

Správnosť je 40/120 = 0,33
Priepustnosť je 40/50 = 0,8

Ak by sme chceli zvýšiť správnosť z 0,33 na 0,5, upravili by sme podmienky vyhľadávania tak, aby sa znížil počet nájdených dokumentov zo 120 na 60. Tým by sa znížil počet relevantných nájdených dokumentov z 40 na 30.

Dostávame:
Správnosť 30/60 = 0,5
Priepustnosť 30/50 = 0,6

Pri tvorení podmienok vyhľadávania musíme určiť pomer medzi správnosťou a priepustnosťou tak, aby sa nám nestalo že nenájdeme veľa relevantných dokumentov alebo nájdeme veľa irelevantných.

Lucene.Net – systém pre indexovanie a vyhľadávanie

Lucene.Net je fulltextový indexujúci a vyhľadávací systém. Bol vyvinutý z projektu Jakarta Lucene implementovaného v Jave a je určený pre jazyk C# na platforme .Net. Poskytuje výkonné funkcie cez jednoduché používateľské rozhranie. Je to Open Source projekt a je šírený pod licenciou (Apache Software License 2.0). Okrem oficiálneho názvu Lucene.Net sa používajú aj DotLucene, dotLucene, Lucene.NET. V tejto dokumentácii budeme tento systém nazývať dotLucene.

Charakteristiky dotLucene [1,11]

- Použiteľnosť v ASP.NET, Win Forms alebo konzolových aplikáciách
- Veľmi dobrý výkon
- Klasifikované výsledky hľadania
- Možnosť zvýrazniť hľadaný výraz vo výsledkoch hľadania
- Prehľadávanie štruktúrovaných aj neštruktúrovaných dát
- Vyhľadávanie v metadátach (podľa dátumu ,hľadanie vo voliteľných poliach indexu a iných)
- Veľkosť indexu je približne 20-30% z veľkosti indexovaného textu
- Vie uchovávať indexované celé dokumenty
- Čistý a jednoduchý kód pod platformou .NET v jednej knižnici (244 KB)
- Licencia Apache Software License 2.0, ktorá umožňuje komerčné aj Open Source používanie
- Lokalizovateľné (Angličtina, Čeština, Francúzština a iné)
- Rozšíriteľnosť (voľný zdrojový kód)
- Možnosť dopĺňať index (pri pridávaní nových dokumentov nie je nutné vytvoriť celý index nanovo)
- Viacindexové vyhľadávanie so spojením výsledkov
- Možnosť indexovať rôzne typy dokumentov (indexovať sa dá akýkoľvek dokument, ktorý sa dá previesť na text)
- Možnosť indexovať dokumenty z rôznych druhov zdrojov (web, databáza, lokálne uložené dokumenty)

- Súčasne je možné spustiť optimalizovanie indexu a vyhľadávanie
- Indexy sú kompatibilné s Lucene pre iné platformy

Hlavné funkcie dotLucene

Nad knižnicou dotLucene boli vytvorené jednoduché nástroje na sprístupnenie poskytovaných funkcií [11]. Popíšme si indexovanie dokumentov a vyhľadávanie kľúčových slov.

Indexovanie

dotLucene poskytuje na indexovanie dokumentov buď triedu, ktorú je možné používať v zdrojovom kóde, alebo jednoduchú konzolovú aplikáciu konfigurovateľnú pomocou XML súboru. Parametre v konfiguračnom súbore sú nasledovné:

- Meno výsledného indexu a cesta k adresáru, v ktorom bude uložený
<index name="index_A" indexFolderUrl="C:\TestIndexA">

- Indexované atribúty
<field name="Nazov" isStored="true" isIndexed="true" isTokenised="false" />

Jednotlivé atribúty majú nasledujúci význam:

- **isStored** - určuje, či bude daný atribút ukladaný do indexu (kedykoľvek sa dá k nemu pristúpiť - je to vhodné pre kratšie texty ako meno autora alebo názov)
- **isIndexed** - určuje, či sa bude dať podľa tohto atribútu vyhľadávať
- **isTokenised** - nastavením toho parametra sa indexovaná časť pred indexovaním rozdelí na slová

- Zdroj dát (databáza, súbor uložený na internetovej adrese alebo lokálnom súborovom systéme)

dotLucene vytvorí index a uloží ho v špecifikovanom adresári, kde je možné k nemu pristupovať a predkladať mu požiadavky na vyhľadávanie v dokumentoch.

Vyhľadávanie

Pri vyhľadávaní dotLucene-u predložíme kľúčové slová, ktoré v indexe vyhľadá. Ako výstup dostaneme zoznam identifikátorov dokumentov, v ktorých sa zadané slová nachádzajú. Taktiež dotLucene vyhodnotí relevanciu nájdených dokumentov.

Uveďme niekoľko základných požiadaviek pre vyhľadávanie dokumentov, v ktorých sa vyskytujú kľúčové slová v istých vzťahoch:

Požiadavka	**Príklad**	**Význam**
Slovo	jablko	Hľadá slovo „jablko“
Výraz	“zelené jablko“	Hľadá výraz „zelené jablko“
V poli	ovocie: jablko	Hľadá slovo „jablko“ v poli „ovocie“
Nahradenie jedného znaku	jablk?	Hľadá slová začínajúce na „jablk“ a končiace ľubovoľným znakom (napr. jablko, jablká)
Nahradenie viac znakov	jablk*	Hľadá slová začínajúce na „jablk“, za ktorými nasleduje ľubovoľný reťazec (napr. jablkový)
Operátor OR	jablko OR med	Alebo – nájde dokumenty obsahujúce slovo „jablko“ alebo „med“
Operátor AND	jablko AND med	A – nájde dokumenty obsahujúce slovo „jablko“ aj slovo „med“
Operátor NOT	NOT jablko	Nájde dokumenty, v ktorých sa nevyskytuje slovo „jablko“
Zoskupovanie	(jablko OR hurška) AND med	Zoskupovanie do zátvoriek
Rozsah	autor: Berg TO Roy	Nájde dokumenty od autorov, ktorí sa abecedne nachádzajú medzi Berg-om a Roy-om
Neurčité hľadanie	jablko~	Hľadanie na základe podobného hláskovania
	jablko~0.9	Požadovaná podobnosť hláskovania
Hľadanie so vzdialenosťou slov	“jablkový koláč”~5	Hľadané slová vo fráze musia byť vzdialené maximálne 5 slov od seba
Relevancia	jablkový ^4 koláč	Slovo „jablkový“ má faktor dôležitosti 4 (prednastavené je 1) – analogicky pre frázy

Tab. 2. Požiadavky na vyhľadávanie v dotLucene

V nasledujúcej tabuľke sú uvedené obmedzenia vyhľadávania.

Požiadavka	**Príklad**	**Význam**
Nahradenie znakov na začiatku výrazu	?ablko *ablko	Chyba: Lucene.Net.QueryParsers.ParseException
Stop slová	and, the, ...	Tieto slová nie sú indexované
Špeciálne znaky +, -, &, |, !, (,), {, }, [,], ~, ^, :, \, ?, *, “	\+	Nutnosť použiť znak ‘\’

Tab. 3. Obmedzenia vyhľadávania v dotLucene

Spracovanie iných ako textových formátov

dotLucene vie priamo spracovávať iba textové formáty a ak chceme spracovávať iné typy súborov, musíme ich previesť do textového formátu [5] alebo ich indexovať pomocou nejakej služby.

Jedným z dostupných nástrojov na indexovanie iných formátov je Seekafile Server [10], ktorý dokáže vytvoriť index z dokumentov vo formáte DOC, PDF, XLS, PPT, RTF, HTML, TXT, XML.

Ďalšou možnosťou je pretransformovať dokument do formátu XML, ktorý je možné indexovať ako je spomenuté v predchádzajúcej časti. Dokumentom, ktoré sa členia na kapitoly, obsahujú nejaké špecifické pole (napr. autor, názov) a pod., môžeme vymyslieť štruktúru, ktorá sa dá previesť do formátu XML.

Použitie dotLucene v našom projekte

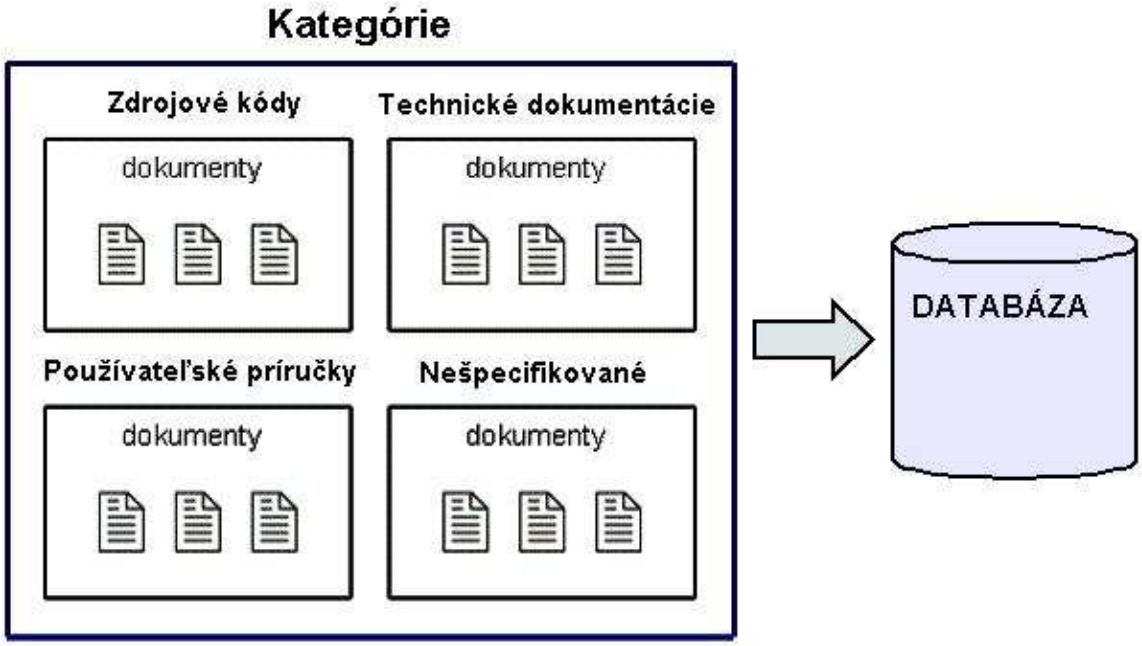
Tento vyhľadávací systém nám ponúka všetky požadované funkcionality (indexácia a vyhľadávanie v rámci databázy alebo lokálnych zdrojov), je výkonný, dá sa používať ako Open Source a ku knižnici je dostupná kompletná API dokumentácia. Preto sa javí ako veľmi dobrý nástroj pre indexovanie a vyhľadávanie v databáze dokumentov a znalostí v našom projekte.

1.5 Kategórie dokumentov

Pri dnešných kapacitných možnostiach databáz je možné uchovávať nesmierne množstvo dokumentov rôzneho druhu. Ak sa používateľ snaží vyhľadať informácie v podobe určitého dokumentu, je veľmi dôležité, aké vyhľadané typy dokumentov mu systém predostrie. Ľahko sa dá predstaviť situácia, kedy by používateľ hľadal informácie podľa určitej skupiny kľúčových slov v dokumente a systém by mu ponúkol dokument, ktorý by síce dané slová obsahoval, ale v úplne inom kontexte.

Týmto problémom sa v súčasnej dobe výskum zaoberá veľmi intenzívne. Viacznačnosť jazyka ako aj slabá reprezentácia znalostí zatiaľ neumožňuje tento problém vyriešiť úplne. Snahou tohto projektu by malo byť eliminovať tento problém čo najviac a poskytovať používateľovi dokumenty poskytujúce informácie, ktoré si z čo najväčšou pravdepodobnosťou vyžiadal.

Aby sa predišlo predkladaniu navzájom nesúvisiacich dokumentov používateľovi, je vhodné dokumenty roztriediť do jednotlivých skupín, čiže kategórií dokumentov. Pod pojmom kategória dokumentu treba rozumieť doplňujúcu informáciu, ktorá prezradí, či sa jedná napríklad o používateľskú príručku, technickú dokumentáciu, dokument zdrojového kódu a podobne (obr. 6). Táto informácia je ďalším kritériom, ktoré môže mať veľký význam pri vyhľadávaní dokumentov. Používateľovi sa budú predkladať iba tie dokumenty, ktoré spolu súvisia čo najviac.



Obr. 6. Schéma rozdelenia dokumentov do kategórií

Matica mier podobnosti medzi kategóriami

Jednotlivé kategórie dokumentov je potrebné zadefinovať a uchovávať medzi nimi určité váhy súvislosti, ktoré budú symbolizovať mieru podobnosti kategórií. Možný spôsob uchovávania váh podobnosti kategórií zobrazuje nasledovná tabuľka.

	Používateľská príručka	Zdrojový kód C#	Zdrojový kód Java	Technická dokumentácia	Nešpecifikovaná kategória
Používateľská príručka	1.0	0.1	0.1	0.3	0.5
Zdrojový kód C#	0.1	1.0	0.75	0.75	0.5
Zdrojový kód Java	0.1	0.75	1.0	0.75	0.5
Technická dokumentácia	0.3	0.75	0.75	1.0	0.5
Nešpecifikovaná kategória	0.5	0.5	0.5	0.5	0.5

Tab. 4. Matica mier podobnosti medzi kategóriami dokumentov

Údaje o vzťahoch medzi jednotlivými typmi dokumentov je vhodné uchovávať v matici, ktorá pomocou indexov zabezpečí rýchly prístup k údajom (tab. 3). Miera podobnosti medzi dvomi kategóriami je vždy rovnaká, matica bude teda symetrická. Medzi rovnakými kategóriami je miera podobnosti 1.0, čiže najvyššia možná. Ostatné miery podobnosti sú v tabuľke zadané iba približne, je ich možné nastaviť a prispôsobiť podľa potreby. Rovnako aj uvedené kategórie a ich počet je len ilustračný.

Ak sa používateľ snaží nájsť iba určitú kategóriu dokumentov, pochopiteľne najviac uprednostňované budú dokumenty práve z tejto kategórie. Na ďalších miestach sa však môžu objaviť aj dokumenty, ktoré sú z inej kategórie blízkej jemu vybranej a obsahujú zadané kľúčové slová. V prípade, že používateľ kategóriu bližšie nešpecifikuje, výber dokumentov sa môže uskutočňovať podľa jeho profilu (analytik, programátor a pod.), prípadne ním často používaných dokumentov. Vtedy sa opäť môžu použiť vzťahy medzi kategóriami.

Maticu, uchovávajúcu miery podobnosti medzi kategóriami, nie je možné vytvárať automatizovane a to z dôvodu, že pri pridaní novej kategórie, počítač nedokáže rozhodnúť, ako sa podobá s ostatnými. Dokonca aj pre rôznych jednotlivcov môžu mať podobnosti kategórií rôznu váhu. Z týchto dôvodov je nutné v prípade novej kategórie jednotlivé miery manuálne zadefinovať človekom (administrátorom) podľa potreby

Automatické zisťovanie kategórie dokumentu

Osobitným problémom je rozpoznanie kategórie dokumentu v prípade, že nie je definovaná. Vtedy by bolo vhodné, aby systém dokázal typ dokumentu zistiť samostatne. To však prestavuje náročnú činnosť. Aby to systém mohol urobiť, musel by vedieť rozoznať kategóriu dokumentu podľa určitých pravidiel, alebo porovnávať dokument z ostatnými a z určitou pravdepodobnosťou kategóriu danému dokumentu priradiť. Je dôležité, aby systém priraďoval kategóriu iba s určitou pravdepodobnosťou, prípadne aby prideľoval miery podobnosti s ostatnými kategóriami. Ak by tak neurobil, riskovalo by sa priradenie zlej kategórie dokumentom, čo by mohlo spôsobiť nežiadané následky.

Spôsobov, akým je možné kategórie zisťovať, je mnoho. Jeden z nich by mohol byť postavený na určitých formách vzorov, podľa ktorých by sa dokumenty kategorizovali. Vytvoriť takýto vzor je veľmi náročné, nakoľko dokumenty spadajúce pod určitú kategóriu sa môžu diametrálne odlišovať. Rovnako by sa na tieto účely dala využiť neurónová sieť, ktorá by sa po predložení veľkého množstva rôznych dokumentov istej kategórie natrénovala a jej výstupom by bola pravdepodobnosť spadania dokumentu pod danú kategóriu. Stále však ostáva problém s dokumentom, ktorý by nespadal ani pod jeden zo spomínaných vzorov kategórie. Systém by mu samozrejme musel priradiť nesprávnu kategóriu, na základe najväčšej podobnosti. Iný spôsob, ktorým by sa dal tento problém obísť, je pri každom dokumente udržiavať všetky pravdepodobnosti, s ktorými dokument spadá do jednotlivých kategórií. Na základe nich by bolo možné vyrátavať vzťahy medzi dokumentmi a predkladať ich používateľovi. Netreba zabúdať na to, že možných kategórií môže byť veľmi veľa a vytváranie pravdepodobnosti podobností je opäť nesmierne náročné.

Automatické zisťovanie kategórie dokumentu je každopádne zložitá činnosť, ktorej úspešná realizácia by však projekt veľmi obohatila. Na automatické zisťovanie by bolo vhodné vytvoriť samostatný nezávislý modul, ktorý by sa dal kedykoľvek vylepšiť, či vymeniť.

1.6 Modelovanie používateľov v znalostnom manažmente

Modelovanie používateľov má svoj vplyv na informačné systémy hlavne z hľadiska nastavenia vlastností a správania sa systému. Vychádzame z faktu, že každý používateľ je jedinečný, má iné očakávania od spôsobu reprezentácie znalostí a od ich parametrov vyhľadávania.
Navyše systémy znalostného manažmentu môžu byť použité aj pre vytváranie nových znalostí (pridanie nových dokumentov), ich prenos a zdieľanie. Systémy takého typu neslúžia iba pre čiru administráciu elektronickej informácie, ale sú schopné učiť sa, napr. procesy používateľov, vzťahy medzi nimi alebo poskytujú štatistiky a rôzne pravidlá pri vyhľadávaní znalostí.

Samotný model pre používateľov je nutný z dvoch dôvodov:

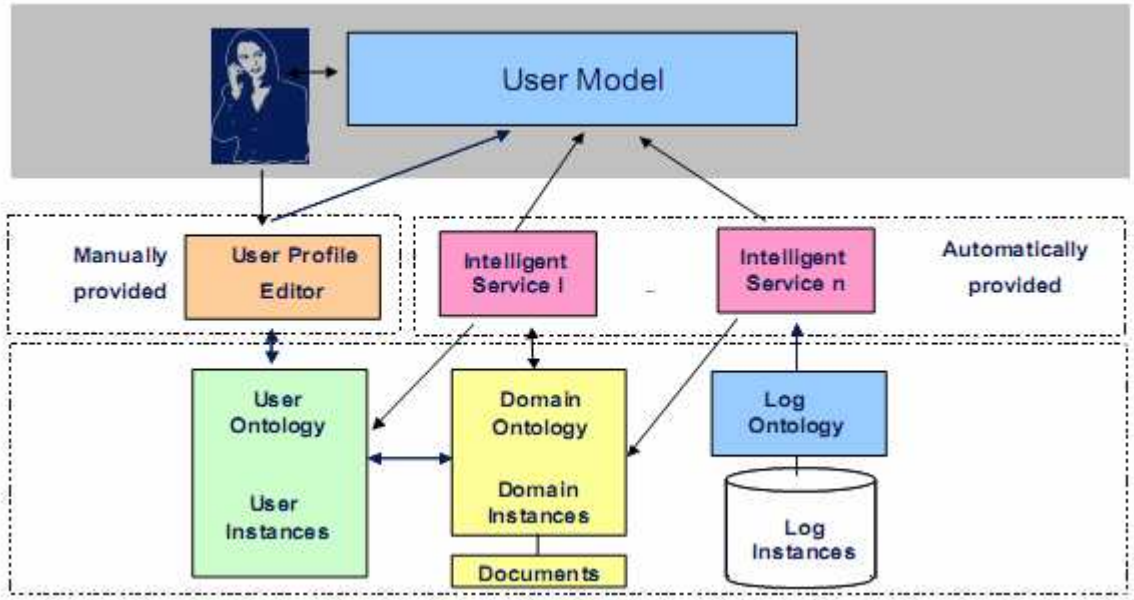
1. Rozdiely medzi jednotlivými individuálnymi potrebami
2. Heterogenita medzi rôznymi skupinami používateľov

Architektúra používateľského modelu založená na ontológiách

Používateľský model je vlastne samostatný modul, ktorý má vlastnosti univerzálneho komponentu

- Adaptácia – podpora konfigurovateľnosti aplikácie na základe charakteristík používateľov, ktoré sú opísané modelom
- Interoperabilita - výmena informácií medzi ďalšími komponentmi
- Znovupoužiteľnosť – opakované použitie v iných systémoch

Bez ujmu na všeobecnosť používateľský model rozdeľujeme na dve časti: manuálne definovaná(explicitná) a automatická časť(implicitná) zabezpečená inteligentnými službami. Manuálna časť je založená na profiloch používateľov, ktoré sa získavajú formou dotazníkov, formulárov. Automatická časť je založená na ďalšom modeli, ktorý obsahuje údaje ktoré sa zbierajú počas interakcie používateľa so systémom, napr. aké typy dokumentov vyhľadával, koľkokrát si ich zobrazil, ktoré navrhnuté dokumenty akceptoval atď. Inteligentné služby majú za úlohu aktualizovať model používateľa a poskytovať služby vzťahujúce sa na práve aktuálneho používateľa. Tu spomenutý koncept integrovania používateľského modelu do systému je zobrazený na obr. 7.[9]



Obr. 7. Model používateľa použitím ontológií

Na obr. 7 sú zobrazené aj tri rôzne ontológie:

- Ontológia používateľov – definuje charakteristiky používateľov a vzťahy medzi nimi
- Ontológia domény – opisuje doménu a koncepty aplikácie
- Ontológia záznamov – definuje sémantiku používateľa so systémom, dáta záznamov sa získavajú monitorovaním tejto interakcie

Pri architektúre sa spomínajú ontológie, ktoré slúžia hlavne pre definovanie domén a vzťahov medzi nimi. Máme však rôzne možnosti ako tieto informácie uchovávať a spracovať. Pri implementácii sa dajú použiť tieto spôsoby:

- Jazyk založený na XML
- Relačná databáza
- Sémanticky silnejšie prostriedky(RDF/OWL)
- Vytvorenými procedúrami rôznych programovacích jazykov(objektovo orientované, procedurálne, logické, ...)

Jazyk založený na XML nám dáva voľnú ruku pri definovaní modelu, čo sa javí ako veľká výhoda, lenže nezabezpečuje univerzálnosť s ďalšími systémami. Relačná databáza už v sebe zahŕňa opis charakteristík a vzťahov medzi nimi, navyše poskytuje prostriedky pre bezpečnosť dát. Databázu v podstate potrebujeme pri ktorejkoľvek reprezentácii modelu.
Ak máme dobre definované charakteristiky používateľov a záznamov interakcie so systémom, môžeme použiť jazyky pre definovanie ontológií. Tieto jazyky sú vytvorené pre definovanie vlastností domén, vzťahov medzi nimi, podmienok a rôznych reštrikcií.

Manuálne definovaná časť modelu používateľa

Na začiatku práce používateľa so systémom ešte nemáme o ňom žiadne informácie, ktoré získame inteligentnými službami. Je žiaduce, aby používateľ poskytol niektoré základné údaje o sebe.

Pretože predpokladáme, že náš systém bude súčasťou nejakého podnikového (enterprise) systému, medzi najdôležitejšie informácie o používateľov by sme zaradili:

- Osobné údaje
- Jeho pozíciu v organizačnej štruktúre
- Aktuálny projekt/y, ktorým/i sa zaoberá
- Skúsenosti a vedomosti
- Oblasti záujmov

Ďalšie údaje, ktoré by sa uchovávali pre jednotlivcov, by mohli byť preferencie týkajúce sa grafického zobrazenia a rozhrania systému. To však neuvažujeme v našom projekte za také dôležité.

Explicitne zadané údaje používateľom sa prejavia pri použití systému - hľadaní znalostí a to hlavne pri ponúknutí alternatívnych možností k výsledkom hľadania získaným iba pomocou kľúčových slov. Z implementačného hľadiska sa táto súčasť najľahšie realizuje pomocou formulárov. Pričom používateľ sa najprv zaregistruje do systému, vytvorí sa mu profil, ktorý si neskôr môže aj modifikovať. Tento spôsob autentifikácie zaručí aj istú úroveň bezpečnosti.

Vytváranie záznamov na základe interakcie používateľa so systémom

Systém znalostného manažmentu sleduje proces jeho požívania jednotlivými používateľmi. Jadro analytického nástroja čerpá z rôznych oblastí výskumu: strojové učenie a štatistika, teóriu agentov a efektívne a rýchle manipulácie s dátami. Na základe poznatkov ako systém pracuje, aký je tok údajov a aké sú dostupné procesy a ako sa pritom používateľ správa, sa vytvorí ontológia záznamov.

Vlastnosťami správania sa môžu byť napr. úroveň aktivity, typ aktivity, zdieľané údaje a štatistické informácie získané pri reakciách na výsledky vyhľadávania.

Inteligentné služby majú zabezpečiť:
- Ponúknuť znalosti podobné aktuálne prezentovanému
- Vybrať znalosti, ktoré zapadajú do oblasti záujmu používateľa, alebo ktoré sú k nemu príbuzné
- Odporučiť členov spoločnosti s podobnými preferenciami

Ak chceme nájsť používateľov, ktorí sú sebe blízky, musíme porovnať ich oblasti záujmu a porovnať ich charakteristiky správania sa. Predpokladajme, že máme dvoch používateľov A a B, skúmame ich „príbuznosť“. Tento vzťah budeme nazývať vzdialenosťou a označovať ako D(A,B). Nech A_d je množina dokumentov, ktoré zobrazil používateľ A pri vyhľadávaní, analogicky aj B_d. Množina $C_{A,B} = A_d \cap B_d = \{Z_1, Z_2, \dots Z_n\}$ je množina dokumentov Z_i, ktoré si vybral aj A aj B. Nech $V_{X,Y}$ označuje hodnotenie dokumentu Y používateľom X, pričom hodnotenie sa zvyšuje pri každom zobrazení daného dokumentu. Vzdialenosť medzi A a B bude vyjadrený vzťahom (4):

$$D(A,B) = \sqrt{\sum_{i}^{C(A,B)} \left(V_{A,Z_i} - V_{B,Z_i}\right)^2} \qquad (4)$$

Teda používatelia A a B sú si blízky ak vyberali podobné dokumenty s približne rovnakou frekvenciou. Čím nižšiu hodnotu dostaneme pri vypočítaní vzdialenosti, tým sú bližší A a B. Najnižšou hodnotou je nula.
Vlastnosť (4) sa dá dobre využiť pri predpovedaní, či sa bude páčiť dokument používateľovi B, na ktorý ešte nehlasoval, ale A už áno. Ak je vzdialenosť používateľov A a B malá, pravdepodobne pri vysokom hodnotení dokumentu používateľom A je logické ponúknuť ju aj druhému. Vzdialenosť dvoch používateľov je však relevantná iba pri vyššom počte prezretých dokumentov.

Vyhľadanie podobných dokumentov je založená na týchto vstupných parametroch: kľúčové slová hľadania, charakteristika používateľa, systémom vytvorené záznamy o jeho aktivite a činnostiach. Najlepším riešením

by bolo dokumenty rozčleniť na rôzne sekcie/kapitoly a potom by sa hľadali kapitoly s takým istým typom. Podobne sa dá jednotlivé dokumenty priradiť projektom, čím zvyšujeme organizáciu v báze znalostí.
Teda sa ľahšie nájdu podobné dokumenty, napr. s rovnakými typmi kapitol, alebo na základe používateľa, ktorý pracuje na podobnom oddelení podniku atď.

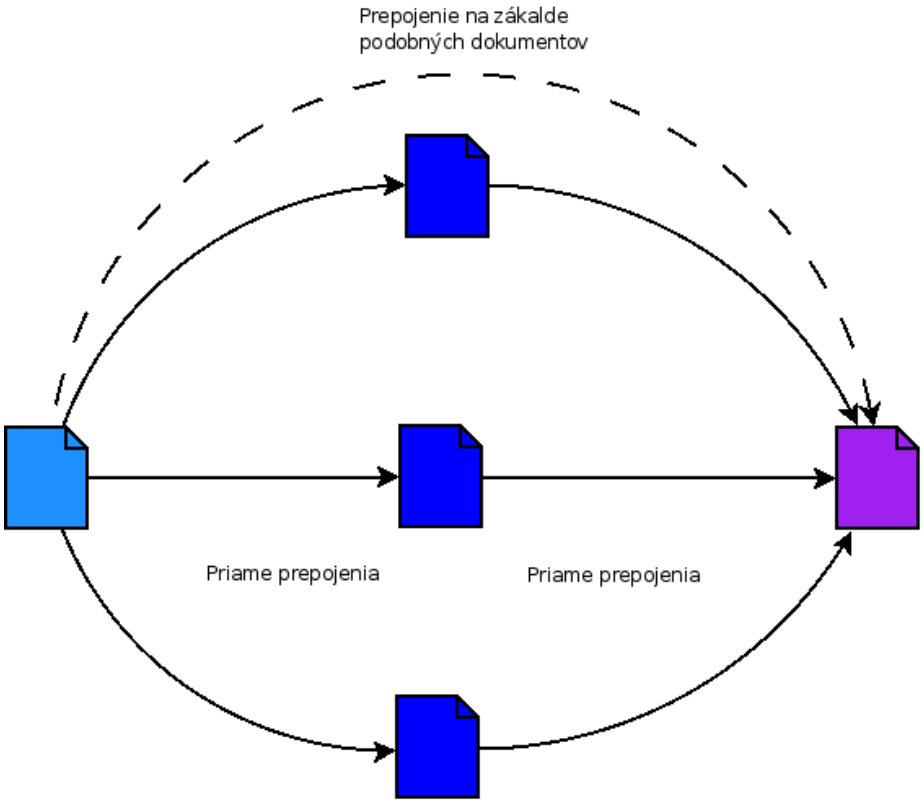
1.7 Určenie relevancie dokumentov založené na topológii grafu

Pri určovaní relevancie dokumentu môžeme vychádzať z niekoľkých možných princípov. Jeden z týchto prístupov vychádza z predpokladu, že dokumenty sú navzájom poprepájané odkazmi a to buď priamymi alebo nepriamymi. Je teda možné nájsť dokument ktorý súvisí s iným dokumentom na základe týchto prepojení.

Výhodou využitia takýchto prepojení je fakt, že sa nám môže podariť nájsť dokument, ktorý síce priamo nesúvisí s nami vyhľadávaným výrazom, ale odkazujú sa naň dokumenty, ktoré s našou požiadavkou priamo súvisia a teda je veľká pravdepodobnosť, že takto nájdený dokument bude relevantný

Princíp spočíva v predstavení si problémového priestoru ako grafu, v ktorom dokumenty tvoria uzly a hrany sú tvorené prepojeniami medzi týmito dokumentmi. Každý z vrcholov je ohodnotený určitým hodnotením, ktoré predstavuje jeho dobrú povesť alebo jeho relevanciu. Čím je táto hodnota vyššia tým je dokument pre nás relevantnejší.

V nasledujúcom texte budú popísané spôsoby ako vytvoriť takýto graf a taktiež možnosti jeho využitia pre potreby nášho projektu.



Obr. 8. Prepojenie dokumentov

Vytváranie grafu

Vytvorenie grafu reprezentujúceho väzby medzi dokumentmi je pravdepodobne najproblematickejšou časťou celého procesu. Obnáša totiž identifikáciu jednotlivých prepojení medzi dokumentmi a taktiež určenie typu týchto prepojení.

Identifikácia uzlov

V princípe existujú dva spôsoby určenia uzlov.

- uzly tvoria dokumenty
- uzly tvoria kapitoly dokumentov

V prípade tvorenia uzlov dokumentmi sa nám značne zjednodušuje situácia pri určovaní spojení medzi nimi, no redukujú sa možnosti pre určenie relevancie jednotlivých spojení. Preto sa ako lepšie riešenie javí použiť ako uzly kapitoly.

Identifikácia prepojení

Prepojenia medzi dokumentmi môžeme klasifikovať do nasledujúcich kategórií.

- Priamy odkaz na dokument: Predstavuje cestu na diskovú jednotku alebo URL dokumentu s ktorým je daný dokument prepojený. Vzhľadom na typ (technická, používateľská dokumentácia, zdrojové kódy) dokumentov, ktorý je v našom projekte spracovávaný je pravdepodobnosť výskytu tohto prepojenia pomerne malá.
- Prepojenie na základe názvov dokumentov a kapitol: Jedná sa o prepojenie v ktorom jeden z dokumentov vo svojom tele obsahuje názov iného dokumentu alebo názov kapitoly obsiahnutej v inom dokumente. Tento typ prepojenia nám umožňuje identifikovať pomerne silne relevantné prepojenia medzi dvoma dokumentmi. Problém môže nastať v prípade negatívneho odkazu na názov (Tento dokument NEsúvisí s dokumentom XY. NErobte to ako je popísané v YZ). Pri použití tohto prepojenia je potrebné uchovávať názvy všetkých dokumentov a názvy všetkých ich kapitol a tie následne porovnávať s textom v dokumente.
- Prepojenie na základe kľúčových slov: Prepoja sa dokumenty, ktoré majú rovnaké kľúčové slová alebo aspoň vysoké percento rovnakých kľúčových slov. Vyžaduje sa vytvoriť zoznam kľúčových slov a porovnávať ho s kľúčovými slovami v danom dokumente.
- Prepojenie na základe výskytu slov: Prepoja sa dva dokumenty ktoré majú vysoký výskyt rovnakých slov.

Všetky tieto prepojenia dokážu určiť súvislosti medzi dvoma dokumentmi. Každému takémuto typu prepojenia môže byť priradená hodnota určujúca jeho dôležitosť, a tak ovplyvniť určenie relevancie dokumentu.

Ďalším spôsobom delenia je delenie na základe toho, či sa jedná o prepojenie medzi kapitolou alebo celým dokumentom. V tomto prípade dokážeme identifikovať tieto druhy prepojení.

1. kapitola - kapitola
2. kapitola - dokument
3. dokument - kapitola
4. dokument - dokument
5. kapitola - kapitola (v rámci dokumentu)

Každému z týchto prepojení môžeme priradiť určitú hodnotu dôležitosti. Napríklad prepojenie medzi dvoma kapitolami môžeme považovať za dôležitejšie ako prepojenie medzi dvoma dokumentmi. Dôležitosť

prepojenia môžeme určovať na základe princípu, v ktorom prepojenie konkrétnejšieho(kapitola) má prednosť pred prepojením všeobecného (celý dokument). Je taktiež podstatné poznamenať že medzi sebou sú prepojené aj kapitoly v jednom dokumente. Toto prepojenie je rovnocenné prepojeniu typu kapitola - kapitola

Prehľadávajúce algoritmy

Pokiaľ existuje graf ktorý máme možnosť prehľadávať je potrebné vybrať vhodný algoritmus, ktorý dokáže určiť hodnoty relevancie jednotlivých uzlov. V našom prípade sa ako vhodný kandidáti javia algoritmy *Pagerank* a *NodeRank*.

PageRank
Algoritmus ohodnocovania vrcholov grafu *pagerank* sa zakladá na jednoduchom princípe, ktorý predpokladá že uzol grafu je tak kvalitný ako vrcholy ktoré sú s ním prepojené. Teda pokiaľ má uzol veľa prepojení s inými uzlami ktoré majú vysoké hodnotenie, tak aj on má vysoké hodnotenie. Princíp je zobrazený na obr. 9.

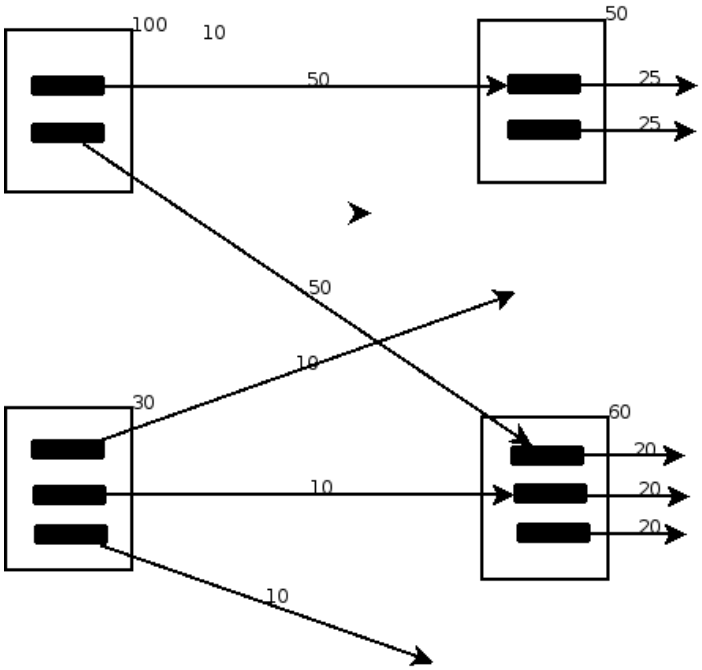
Pri postupnom prechode cez všetky uzly však môže dôjsť k situácii keď sú uzly opakovane ohodnocované a tak môže dôjsť k situácii keď hodnotenie niektorých uzlov rastie do nekonečna, a dôjde k takzvanému "zahlteniu ohodnotenia".

Pre odstránenie tohto prípadu je používaná stratégia náhodného chodca (*random walker*). Pri použití tejto stratégie sa prechádza grafom a s určitou pravdepodobnosťou sa preskočí do iného uzla. Týmto spôsobom dokážeme predísť zahlteniu.

NodeRank
Na rozdiel od algoritmu *PageRank* dokáže pracovať s grafom ktorého hrany sú ohodnotené. Pracuje na podobnom princípe ako *PageRank*. Teda posúva hodnotenie uzlu grafu do jeho nasledovníkov, no využíva pri tom aj hranu ktorá tieto dva uzly spája. Teda hodnotenie nasledovníka uzla je tvorené jednak predchodcom ohodnotením uzla samotného ako aj váhou hrany ktorá uzol a jeho nasledovníka spája. V prípade rozhodovania do ktorého ďalšieho uzlu sa pri prechode grafom vyberieme slúži váha prepojení ktoré z uzla vychádzajú.

V algoritme je taktiež použitá stratégia náhodného chodca. Táto stratégia je však modifikovaná tým spôsobom, že v prípade návštevy toho istého vrcholu niekoľkokrát po sebe, dostáva vrchol stále menší a menší prírastok ohodnotenia.

Pre potreby nášho projektu sa ako výhodnejšie javí použitie algoritmu *NodeRank*, ktorý berie do úvahy aj váhy prepojení medzi dokumentmi.



Obr. 9. Princíp ohodnocovania vrcholov algoritmom PageRank

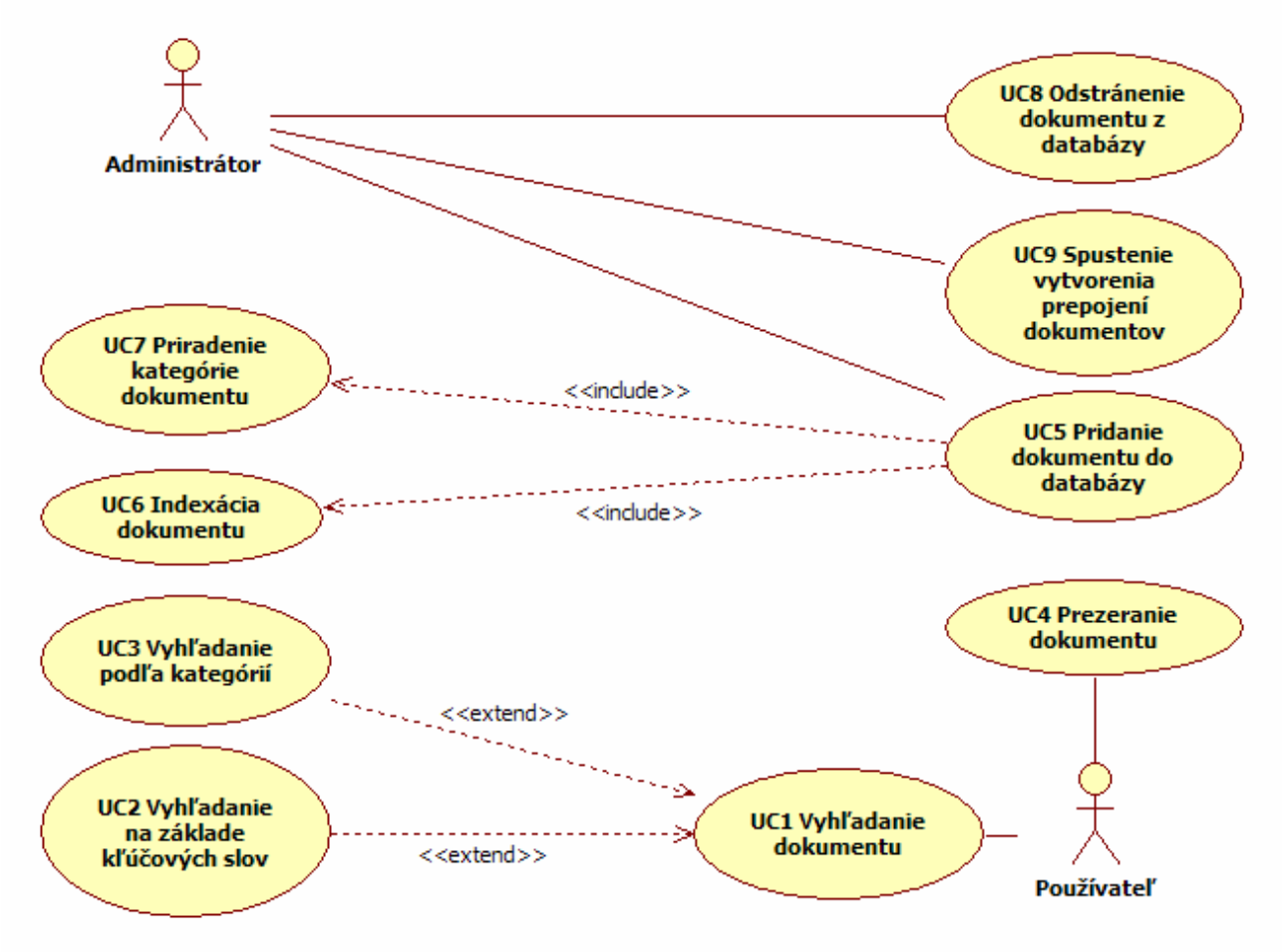
Vhodnosť algoritmov

Pre potreby nášho projektu sa ako vhodnejšie javí použitie algoritmu *NodeRank* vzhľadom na to že dokáže pracovať aj s váhami medzi jednotlivými uzlami v grafe. Je taktiež nutné využitie stratégie náhodného chodca kvôli predídeniu zahlteniu ohodnotenia.

2 Špecifikácia

2.1 Špecifikácia požadovaného riešenia

Na obr. 10 je diagram prípadov použitia, v ktorom sú znázornené funkcionality systému poskytnuté jednotlivým hráčom.



Obr. 10. Diagram prípadov použitia

2.2 Identifikácia hráčov

Systém bude používaný dvomi kategóriami hráčov.

Administrátor - bude spravovať databázu, bude do nej pridávať nové dokumenty, priraďovať kategóriu dokumentu, odstraňovať dokumenty. Po pridaní nových dokumentov do databázy je potrebné aktualizovať bázu znalostí, tak aby sa neporušila konzistencia dát. Keďže tieto operácie môžu byť náročné na čas a počas ich vykonávania sa nesmie s databázou pracovať, spúšťať ich bude administrátor (napr. v noci).

Používateľ - je mu umožnené vyhľadávať dokumenty v systéme. Môže zadať kľúčové slová alebo kategóriu dokumentu, podľa ktorých znalostný systém vyhodnotí vhodnosť dokumentov pre požiadavku používateľa. Následne mu systém poskytne zoznam kandidátov, ktorých

vyhodnotil ako najvhodnejších a používateľ si môže podľa krátkej časti textu dokumentu vybrať ten dokument, ktorý má tendenciu byť skutočne relevantným vzhľadom na jeho požiadavky.

Keďže jeden modul v systéme bude analyzovať aké dokumenty používateľ často hľadá, resp. používa, bude vedená evidencia používateľov. Ak sa používateľ prihlási ako už evidovaný používateľ, systém na základe podobnosti hľadaní a výberov dokumentov s inými používateľmi poskytne výbery iných používateľov, ktoré by mohli byť pre hľadajúceho zaujímavé. Ak sa používateľ nehlási ako evidovaný, táto funkcionalita systému bude nefunkčná.

2.3 Prípady použitia

Prípady použitia sú popísané v nasledujúcich tabuľkách.

Identifikátor	UC1		
Názov	Vyhľadanie dokumentu		
Opis	Používateľ vyhľadá dokument podľa svojich preferencií		
Priorita	vysoká	**Frekvencia**	Denne, rádovo stovky
Vstup. podm.	Požiadavka na vyhľadanie dokumentu		
Výstup. podm.	Zobrazenie výsledkov vyhľadávania		
Používatelia	Používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ: Spustenie grafického rozhrania programu	
	2	Používateľ: Výber typu vyhľadávania	
	3	Používateľ: Zadanie kľúčových slov pre vyhľadávanie (UC2) a/alebo kategórie vyhľadávania(UC3)	
	4	Systém: Spracovanie požiadavky	
	5	Systém: Zobrazenie výsledkov vyhľadávania používateľovi	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Používateľ: Zrušenie vyhľadávania	

Identifikátor	UC2		
Názov	Vyhľadanie na základe kľúčových slov		
Opis	Používateľ vyhľadá dokument podľa kľúčových slov		
Priorita	vysoká	**Frekvencia**	Denne, rádovo stovky
Vstup. podm.	Zobrazenie formulára na zadanie kľúčových slov		
Výstup. podm.	Zobrazenie výsledkov vyhľadávania		
Používatelia	Používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ: Zadanie kľúčových slov	
	2	Systém: Spracovanie požiadavky	
	3	Systém: Zobrazenie výsledkov vyhľadávania používateľovi	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Používateľ: Zrušenie vyhľadávania	

Identifikátor	UC3		
Názov	Vyhľadanie podľa kategórií		
Opis	Používateľ vyhľadá dokument podľa kategórií, do ktorých dokument patrí		
Priorita	vysoká	**Frekvencia**	Denne, rádovo stovky
Vstup. podm.	Zobrazenie formulára na zadanie kategórií		
Výstup. podm.	Zobrazenie výsledkov vyhľadávania		
Používatelia	Používateľ		

Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ: Zadanie kľúčových slov
	2	Systém: Spracovanie požiadavky
	3	Systém: Zobrazenie výsledkov vyhľadávania používateľovi
Alternatívna postupnosť	**Krok**	**Činnosť**
	-	Používateľ: Zrušenie vyhľadávania

Identifikátor	UC4		
Názov	Prezeranie dokumentu		
Opis	Používateľ si prezrie dokument, ktorý mu poskytol systém vo výsledkoch		
Priorita	vysoká	**Frekvencia**	Denne, rádovo stovky
Vstup. podm.	Zobrazené výsledky vyhľadávania		
Výstup. podm.	Zobrazenie zvoleného dokumentu		
Používatelia	Používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Systém: Zobrazenie výsledkov vyhľadávania	
	2	Používateľ: Výber dokumentu, ktorý chce používateľ prezerať	
	3	Systém: Zobrazenie zvoleného dokumentu	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Používateľ: Zrušenie zobrazovania výsledkov	

Identifikátor	UC5		
Názov	Pridanie dokumentu do databázy		
Opis	Administrátor pridá nový dokument do databázy		
Priorita	vysoká	**Frekvencia**	Mesačne, rádovo desiatky
Vstup. podm.	Nový dokument		
Výstup. podm.	V databáze je uložený nový dokument, jeho meno, metadáta.		
Používatelia	Administrátor		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Administrátor: Požiadavka na systém, aby zobrazil formulár pre pridanie dokumentu do databázy	
	2	Systém: Zobrazí administrátorovi formulár na pridanie dokumentu do databázy	
	3	Administrátor: Zadá cestu k dokumentu	
	4	Systém: Uloží do databázy meno dokumentu, binárne dáta, a metadáta vo formáte XML (okrem iného musia byť úspešne vykonané UC6 a UC7)	
	5	Systém: Zobrazí administrátorovi správu o úspešnom vykonaní operácie	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Administrátor: Zrušenie pridávania dokumentu do databázy	
	4.a	Systém: Dokument nebude uložený do databázy	
	5.a	Systém: Vypísanie dôvodu neúspešného pridania dokumentu do databázy	

Identifikátor	UC6		
Názov	Indexácia dokumentu		
Opis	Systém pridelí dokumentu jedinečné identifikačné číslo a prepíše index		
Priorita	vysoká	**Frekvencia**	Mesačne, rádovo desiatky
Vstup. podm.	Existujúci dokument		
Výstup. podm.	Pridelený identifikátor dokumentu, prepísaný indexu		
Používatelia			

Základná postupnosť	**Krok**	**Činnosť**
	1	Systém: Pridelenie identifikátora novému dokumentu
	2	Systém: Aktualizácia indexu
	3	Systém: Zobrazenie správy o úspešnosti vykonanej operácie

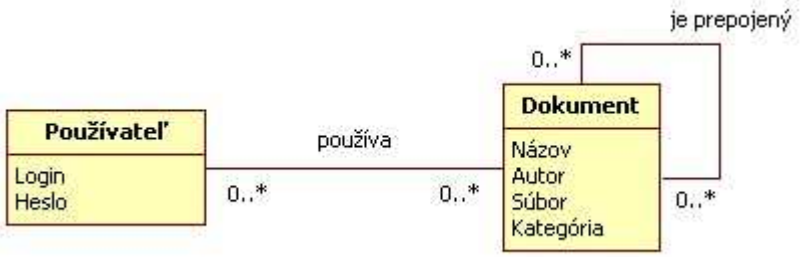
Identifikátor	UC7		
Názov	Priradenie kategórie dokumentu		
Opis	Administrátor		
Priorita	stredná	**Frekvencia**	Mesačne, rádovo desiatky
Vstup. podm.	Existujúci dokument		
Výstup. podm.	Priradená kategória dokumentu		
Používatelia	Administrátor		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Systém: Poskytnutie rozhrania pre zadanie kategórie dokumentu	
	2	Administrátor: Zadanie kategórie (existuje aj kategória „bez kategórie“)	
	3	Systém: Priradenie zadanej kategórie dokumentu	
	4	Systém: Zobrazenie správy o úspešnosti vykonania operácie	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Administrátor: zrušenie priradenia kategórie (systém priradí kategóriu „bez kategórie“)	

Identifikátor	UC8		
Názov	Odstránenie dokumentu z databázy		
Opis	Administrátor		
Priorita	vysoká	**Frekvencia**	Ročne, rádovo jednotky
Vstup. podm.	Dokument uložený v databáze		
Výstup. podm.	Dokument sa v databáze ďalej nenachádza, je odstránený z indexu aj tabuliek jednotlivých modulov		
Používatelia	Administrátor		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Systém: Poskytnutie rozhrania pre výber dokumentu na odstránenie	
	2	Administrátor: Zvolenie dokumentu, ktorý má byť odstránený	
	3	Systém: Odstránenie dokumentu z databázy	
	4	Systém: Odstránenie všetkých výskytov odstráneného dokumentu z indexu a zo všetkých tabuliek v databáze	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Administrátor: zrušenie odstránenia dokumentu	

Identifikátor	UC9		
Názov	Spustenie vytvorenia prepojení dokumentov		
Opis	Administrátor		
Priorita	vysoká	**Frekvencia**	Mesačne, rádovo desiatky
Vstup. podm.	Databáza dokumentov, pridané nové dokumenty do databázy		
Výstup. podm.	Znalostným systémom vytvorené požadované prepojenia medzi novými a starými dokumentmi v databáze		
Používatelia	Administrátor		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Administrátor: Spustenie operácie	
	2	Systém: Modul pracujúci s node-rankingom vytvorí prepojenia medzi novými dokumentmi a ostatnými dokumentmi uloženými v databáze	
	3	Systém: Informovanie o úspešnosti vykonania operácie	

2.4 Model údajov

Model údajov použitý v systéme sa nachádza na obr. 11.



Obr. 11. Logický model údajov

Entity modelu údajov:

Dokument – predstavuje dokument uložený v databáze. Pozostáva z názvu, autora, kategórie a samotného dokumentu v binárnych dátach (poskytnutého používateľovi v prípade výberu daného dokumentu). Dokumenty sú medzi sebou poprepájané (priame odkazy, podobnosť kľúčových slov, podobnosť názvov kapitol, podobnosť indexovaného obsahu dokumentu).

Používateľ – vystupuje v systéme ako niekto, kto vyhľadáva dokumenty. Evidovaný používateľ má svoje meno, ktoré jednoznačne určuje jeho identitu. Systém si uchováva informácie o tom, ktorý dokument používateľ vyhľadá a/alebo použije a poskytne mu dokumenty, ktoré vyberal iný používateľ s podobným výberom.

2.5 Nefunkcionálne požiadavky

Systém bude implementovaný v jazyku C# na platforme .NET verzia 2.0. Aplikácia bude poskytovať grafické používateľské rozhranie, ktoré umožní používateľovi vyhľadávať v báze znalostí a administrátorovi pridávať a odstraňovať dokumenty z databázy, pričom databáza bude relačná.

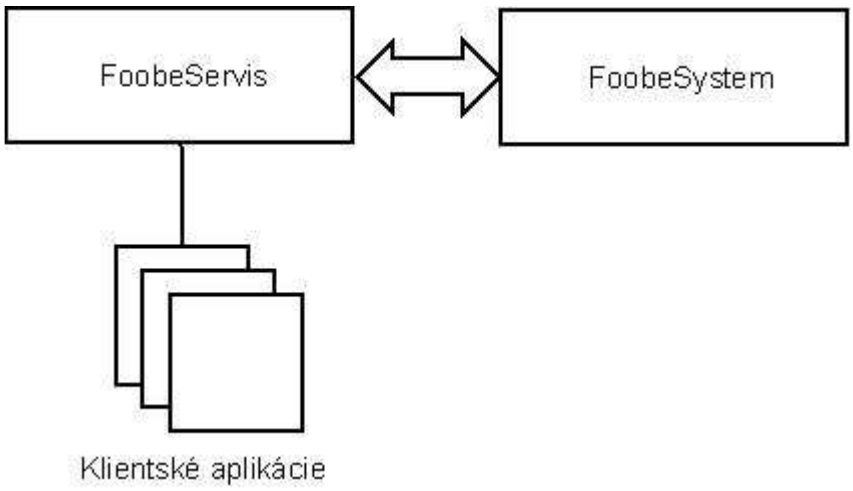
3 Návrh

Systém bude založený na architektúre klient – server.

Server bude poskytovať webovú službu v triede *FoobeServis*. Táto služba bude spolupracovať s databázou popísanou v kapitole 3.2. Znalostný manažment bude spravovaný systémom, ktorého architektúre sa venuje nasledujúca kapitola.

Klientská aplikácia poskytne rozhrania pre administráciu znalostného manažmentu a používateľov, ktorí budú služby tohto manažmentu používať. Tieto rozhrania budú volať metódy, ktoré poskytne webová služba bežiaca na serveri.

Návrh architektúry systému je na obr. 12.



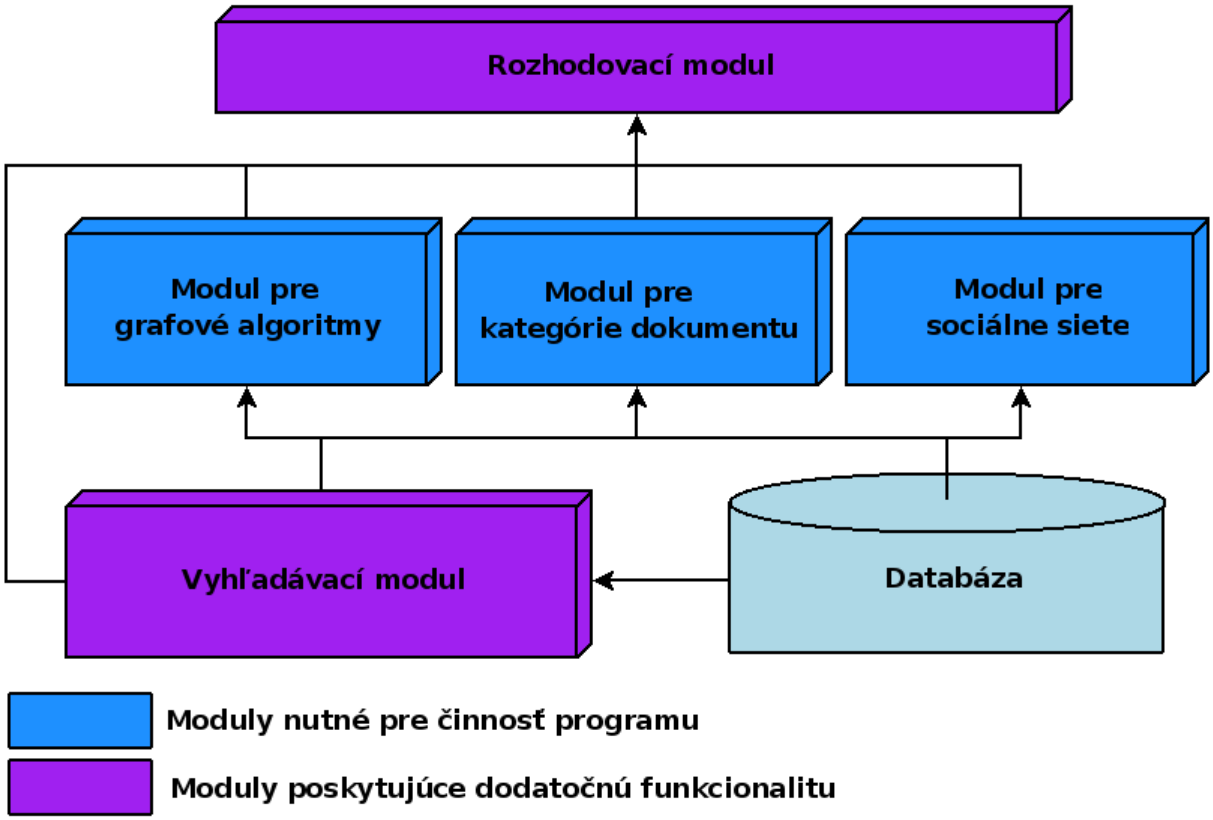
Obr. 12. Architektúra systému

3.1 Architektúra systému pre manažment znalostí (FoobeSystem)

V nasledujúcej kapitole bude popísaná architektúra systému, jeho časti a spôsob prepojenia medzi nimi (obr. 12).

Časti z ktorých s systém skladá môžeme rozdeliť do nasledujúcich kategórií:

1. Databáza
2. Moduly nutné k činnosti programu
3. Moduly poskytujúce dodatočnú funkcionalitu



Obr. 13. Architektúra systému pre manažment znalostí

Databáza

V databáze sú uchovávané všetky informácie potrebné pre činnosť systému a jeho jednotlivých modulov. Znamená to, že sú tu uchovávané údaje ako dokumenty, prepojenia medzi dokumentmi a ich kategóriami, zoznamy toho kedy kto použil ktorý dokument.

Moduly nutné pre činnosť systému

Sú to moduly ktoré poskytujú svoje výstupy iným modulom alebo priamo používateľovi. Vzhľadom na fakt že ostatné moduly sú závislé na výstupe týchto modulov , funkčnosť systému v prípade vynechanie jedného z nich nie je možná.

V prípade existencie len týchto modulov, je systém schopný poskytovať výstupy, no ich kvalita je otázna, lebo pôjde len o výstupy na základe vyhľadávanie zhody slov v texte.

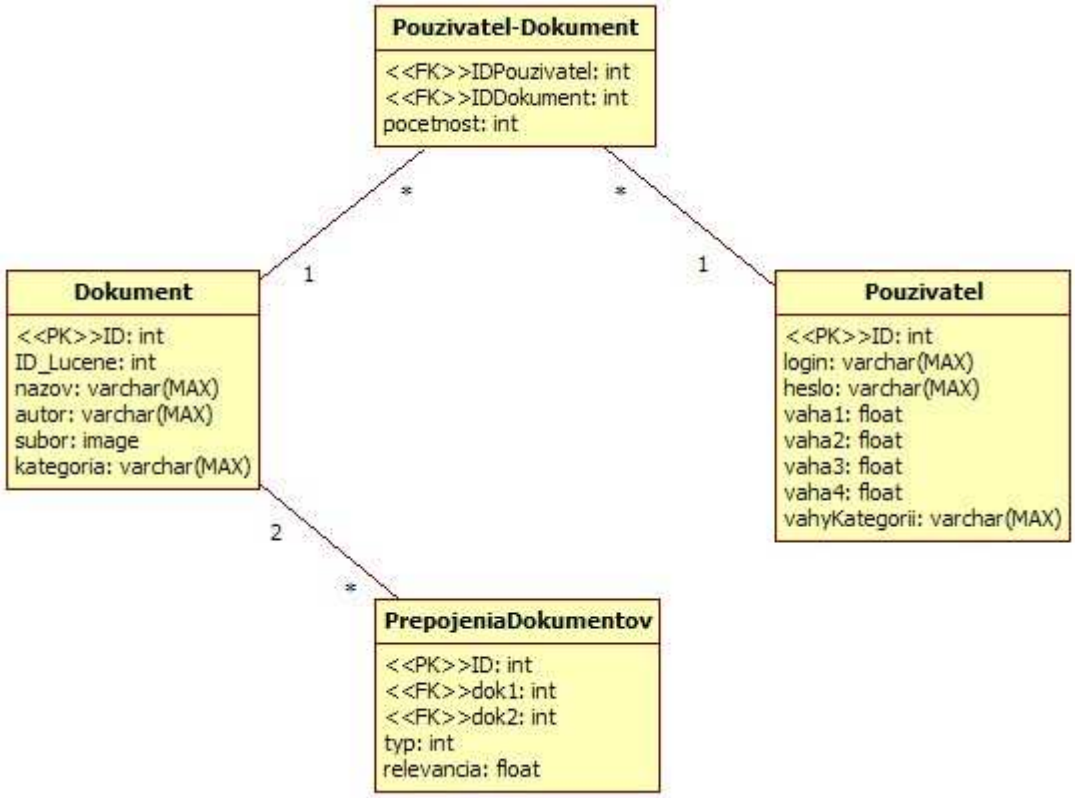
Moduly poskytujúce dodatočnú funkcionalitu

Tieto moduly, ako je zrejmé z názvu poskytujú dodatočnú funkcionalitu pre vyhľadávanie znalostí. Ich použitím bude vyhľadávanie znalostí obohatené o výstupy ktoré b pri použití len základných modulov neboli nájdené.

3.2 Databáza

Na uchovávanie údajov sa použije databáza MS SQL Server 2005. Bude uložená na serveri spolu so systémom *Foobe*, ktorý bude prístupný ako webová služba v triede *FoobeServis*.

V databáze budú uložené údaje o dokumentoch a údaje jednotlivých modulov potrebné pre ich funkčnosť. Na obr. 14 je fyzický model údajov uložených v databáze.



Obr. 14. Fyzický model údajov uložených v databáze

Dokument - atribútmi tabuľky sú názov dokumentu, autor dokumentu, kategória dokumentu, atribút súbor reprezentuje binárnu formu súboru, ktorá bude poskytnutá používateľovi pri požiadavke na prezeranie

Pouzivatel - tabuľka evidovaných používateľov, ktorí majú jedinečné prihlasovacie mená; okrem prihlasovacieho mena sa eviduje aj heslo a nastavenia váh, ktoré budú používať moduly systému

Pouzivatel-Dokument - tabuľka uchováva údaje o tom, ktoré dokumenty používatelia prezerali

PrepojeniaDokumentov - údaje o prepojeniach dokumentov podľa algoritmu ohodnocovania uzlov (*NodeRank*)

3.3 Vyhľadávací modul

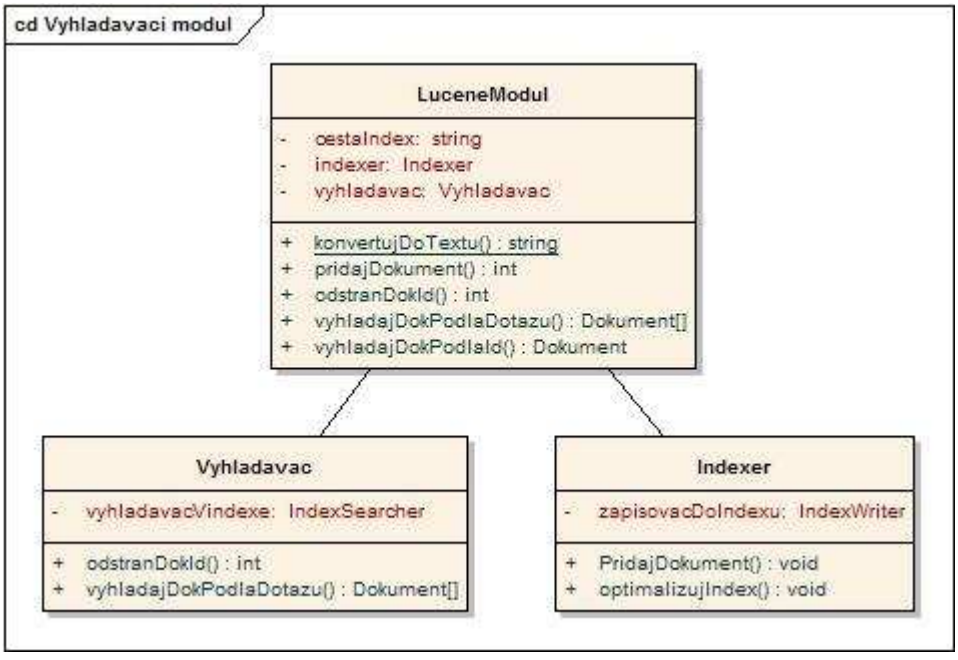
Tento modul bude pevnou súčasťou celého systému. Keďže vyhľadávanie podľa kľúčových slov je základom pre dolovanie v dátach, modul bude poskytovať indexáciu dokumentov uložených v databáze a vyhľadávanie zadaných slov v indexovaných dokumentoch.

Je potrebné uvedomiť si, že dokumenty musia byť jednoznačne identifikovateľné vo všetkých moduloch. V databáze budú ukladané pod jednoznačným identifikátorom. Tento jednoznačný identifikátor bude vygenerovaný a priradený dokumentu pri jeho indexácii, následne sa pridá do databázy.
Vyhľadávací modul eviduje k dokumentu položky:

- *názov* dokumentu
- *autor* dokumentu
- *subor* (názov súboru)
- *kategoria* (kategória priradená dokumentu pre zdokonalenie vyhľadávania).

Tieto položky budú nielen indexované, ale zároveň aj uložené v indexe tohto modulu. Samotný obsah dokumentu bude najskôr prevedený do textovej formy (bude obsahovať iba čistý text dokumentu, a nebude obsahovať značky používané pre formátovanie súboru), potom bude indexovaný štandardným „parserom“, ktorý poskytuje dotLucene. Binárne dáta dokumentov sú uložené v databáze.

Modul bude pozostávať z inštancií dvoch tried: *Indexer* a *Vyhladavac*. Diagram tried celého modulu je znázornený na obrázku nižšie.



Obr. 15. Diagram tried vyhľadávacieho modulu

Trieda *LuceneModul*:
Hlavná trieda modulu. Eviduje cestu k indexu, s ktorým sa pracuje. Pomocou inštancií tried *Indexer* a *Vyhladavac* s ním pracuje, tzn. pridáva dokumenty, odstraňuje dokumenty.

Trieda *Indexer*:
Táto trieda obsahuje objekt triedy *IndexWriter*, ktorý poskytuje funkcie pre zapisovanie do indexu. V metóde *PridajDokument(string, string, string, string, string, string)* zavolá funkciu *AddDocument* triedy *IndexWriter*, ktorá zapíše nový dokument do indexu. Konverziu dokumentu do čistého textu prevedie statická funkcia *konvertujDoTextu(string)* implementovaná v triede *LuceneModul*.

Pre optimalizáciu indexu (spojenie verzií indexov a aktualizáciu) sa použije metóda *optimalizujIndex()*.

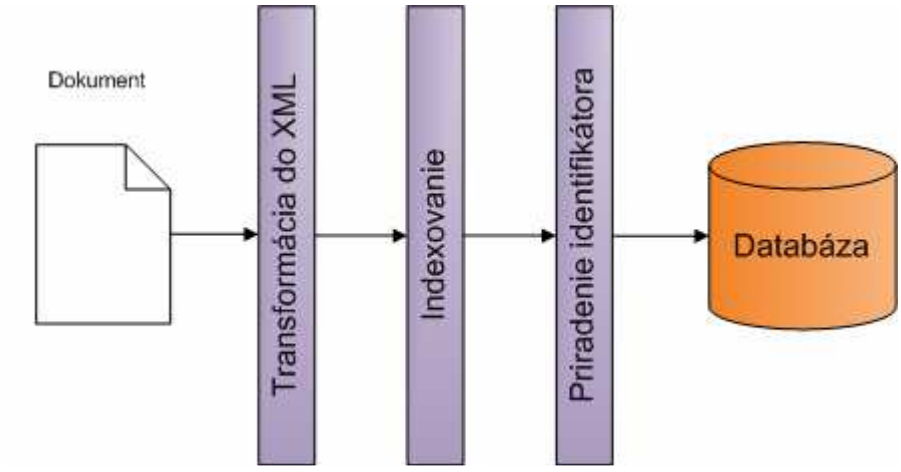
Trieda *Vyhladavac*:
Použije sa na narábanie s indexom na úrovni vyhľadávania, s ktorým súvisí aj odstraňovanie dokumentov v indexe. Vytvorí inštanciu triedy *IndexSearcher*, ktorá vyhľadáva v dokumente podľa zadaných požiadaviek.

Metódy na vyhľadávanie bude nasledovná: *vyhladajDokPodlaDotazu(string)*. Na vstup dodáme vyhľadávací dotaz podľa syntaxe systému na indexáciu a vyhľadávanie dotLucene. Metóda vráti pole objektov štruktúry *Dokument*, ktorá je bližšie popísaná v časti *Vyhľadávanie dokumentov podľa kľúčových slov*.

Pridanie dokumentu do databázy

Oproti pôvodnému návrhu sa zmenil koncept súboru informácií, ktoré budeme o dokumente evidovať. Pôvodne mali byť špecifické časti dokumentu zapísané do súboru v XML formáte, kde by bolo jednoznačne jasné, čo v danom atribúte evidujeme. Mohli to byť kapitoly, podkapitoly, bibliografia, autor, názov, a iné položky, ktoré je možné evidovať pri dokumentoch. Avšak pre tieto účely by bolo možné evidovať iba dokumenty štruktúrované v dohodnutom formáte.

Proces pridania dokumentu do databázy by bol trojstupňový. Znázornený je na obr. 16.



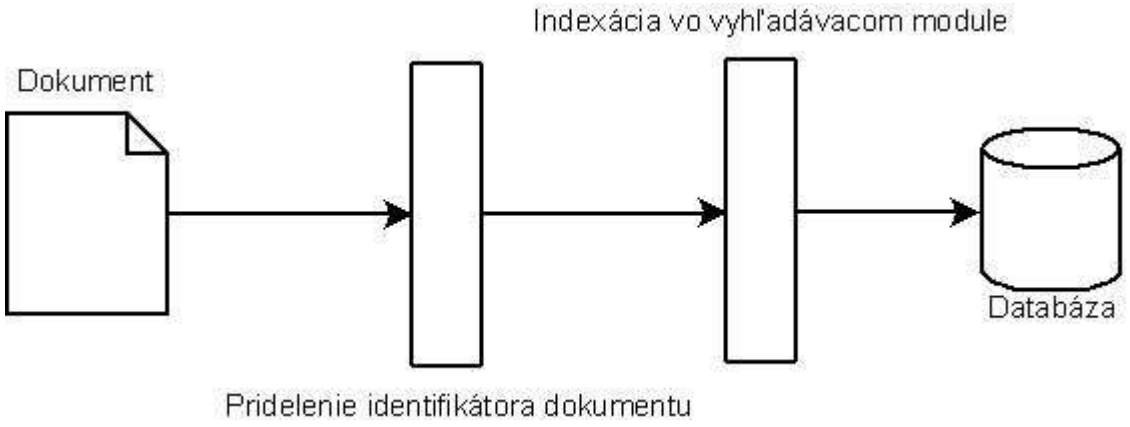
Obr. 16. Pridanie dokumentu a XML transformácie do databázy

Najskôr by bol dokument pretransformovaný do formátu XML. dotLucene by z tejto reprezentácie pridal potrebné informácie do indexu dokumentov, pričom by vytvoril dokumentu jedinečný identifikátor, ktorý by

sa použil ako identifikátor dokumentu v databáze. Potom by boli dokument, jeho XML forma a názov uložené do databázy.

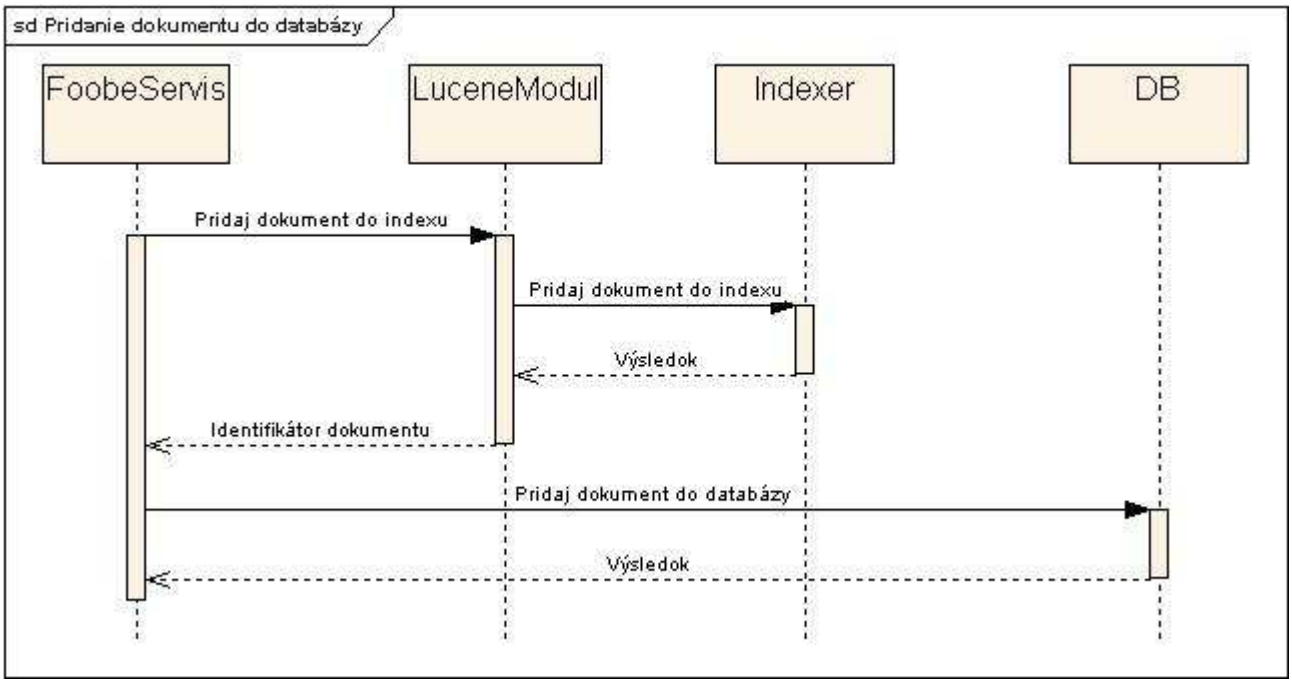
Kvôli použiteľnosti systému pre vyhľadávanie znalostí v dokumentoch bez predpísanej štruktúry však bol tento prístup zamietnutý. Nová koncepcia umožní použitie systému v rozsiahlejšom priestore dokumentov. Dokumenty nebudú členené na špecifické časti a transformované do súboru vo formáte XML, ale indexovaný bude celý obsah dokumentu.

Pridanie dokumentu do databázy je znázornené na obr. 17.



Obr. 17. Pridanie dokumentu do databázy

Najskôr dokument poskytneme vyhľadávaciemu modulu, ktorý ho spracuje a vytvorí dokumentu jedinečný identifikátor. Tento identifikátor sa použije ako identifikátor dokumentu v databáze. Následne bude samotný dokument s príslušnými údajmi uložený do databázy. Na nasledujúcom obrázku je sekvenčný diagram pridania dokumentu do databázy.



Obr. 18. Sekvenčný diagram pridania dokumentu do databázy

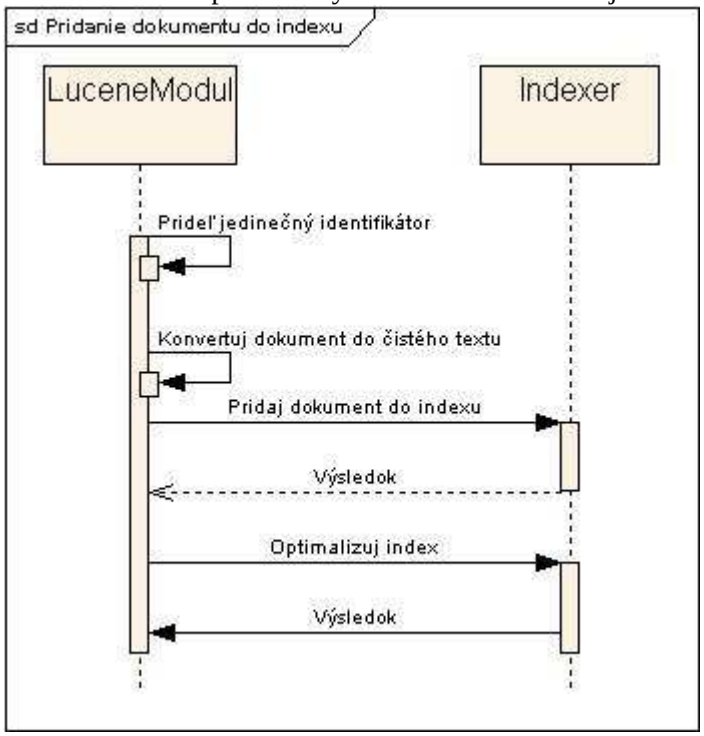
Spracovanie dokumentu vyhľadávacím modulom prebieha v štyroch fázach:

1. pridelenie identifikátora novému dokumentu
2. transformácia do čistého textu
3. indexácia
4. optimalizácia indexu

Na transformáciu dokumentu do čistého textu sa použijú triedy *RtfPlugin* pre konverziu dokumentov vo formáte *RTF* a trieda *IFilter* pre dokumenty formátu *DOC, DOCX, XLS, XLSX, PDF, PPT, HTM, HTML, TXT*. Obe triedy sa nachádzajú v projekte *Seekafile.Plugin.Default*. Volanie metód týchto tried, ktoré prevedú obsah dokumentov na čistý text sa prevedie v statickej metóde *konvertujDoTextu(string cestaKsuboru)* triedy *LuceneModul*.

Indexáciu umožní trieda *Indexer* v metóde *PridajDokument(string id, string nazov, string autor, string kategoria, string subor, string cestaKsuboru)*. Pre dokončenie indexácie sa zavolá metóda *optimalizujIndex()*, ktorá aktualizuje používaný index, čím je možné vyhľadávať práve pridaný dokument.

Sekvenčný diagram pridania dokumentu z pohľadu vyhľadávacieho modulu je na obr. 19.



Obr. 19. Sekvenčný diagram pridania dokumentu do indexu

Odstraňovanie dokumentu z databázy

Pri odstraňovaní dokumentu z databázy musí systém odstrániť nielen samotný dokument vrátanie XML formátu, názvu a kategórie v databáze, ale aj všetky prepojenia vo všetkých moduloch a všetky výskyty v indexoch. Aby sme mohli odstrániť tieto doplňujúce údaje z bázy dát, musíme poznať identifikátor referencií, ktorým je identifikátor dokumentu pridelený pri pridávaní do databázy.

V dotLucene je implementovaných niekoľko spôsobov na odstraňovanie dokumentov z indexov. Napr. trieda *IndexModifier*, ktorá je súčasťou balíka *Lucene.Net.Index*, poskytuje funkciu *DeleteDocument*, ktoré z indexov odstránia všetky výskyty dokumentu, ktorého identifikátor zadáme ako argument funkcii.

Vyhľadanie dokumentov podľa kľúčových slov

Je zrejmé, že najdôležitejšou úlohou tohto modulu je vyhľadávanie dokumentov na základe kľúčových slov. Knižnice dotLucene ponúkajú kompletnú podporu vyhľadávania v indexoch. Je preto veľmi jednoduché nájsť všetky dokumenty, ktoré obsahujú zadané kľúčové slová. Tieto funkcie poskytuje trieda *Net.Lucene.Search.IndexSearcher*.

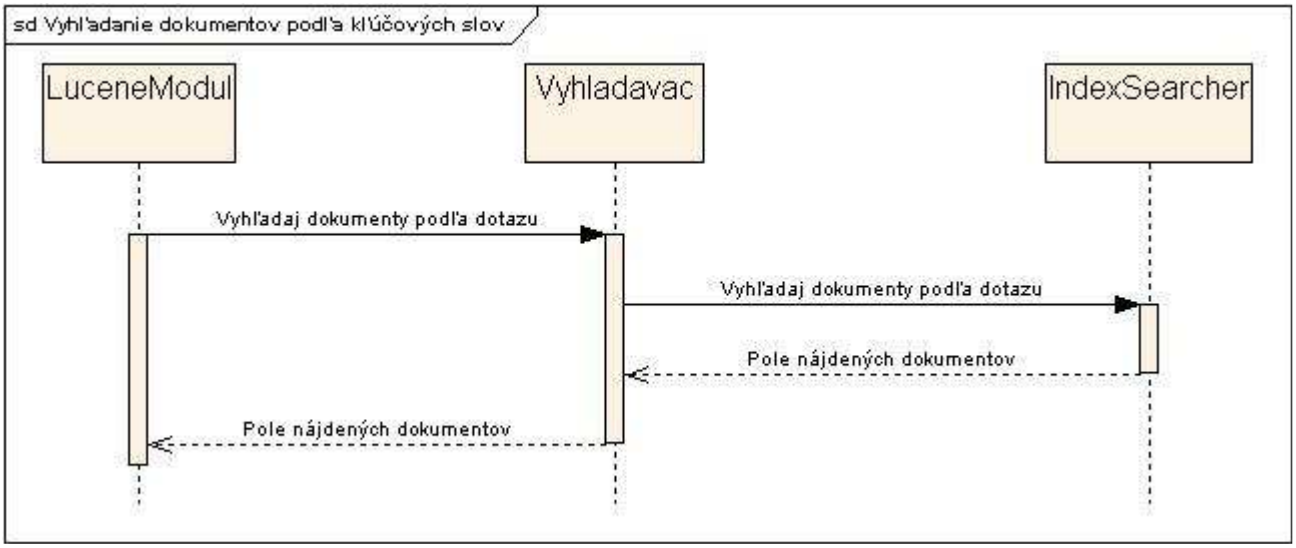
Vyhľadávací modul na vyhľadávanie používa metódu:

public Hits search(Query query, Sort sort)

- Metóda vracia množinu nájdených dokumentov (jedna položka obsahuje identifikátor dokumentu, relevanciu, a pod.), ktorou je možné prostredníctvom iterátora prechádzať výsledkami zoradenými podľa relevancie.
- Argumentmi sú požiadavka (na úpravu z textového vstupu zadaného používateľom do formátu očakávaného funkciou použijeme triedu *QueryParser*) a metóda usporiadania výsledkov (podľa relevancie alebo podľa usporiadania v indexe). Modul používa usporiadanie podľa relevancie (RELEVANCE).

V prípade potreby je možné použiť metódu *public Hits search(Query query, Filter filter, Sort sort)*, ktorá medzi výsledky zaradí iba dokumenty splňujúce podmienky dané argumentom *filter*.

Výsledky budú spracované hlavným vyhodnocovacím systémom, ktorý ich poskytne používateľovi a ponúkne prezretie najvhodnejších dokumentov. Sekvenčný diagram vyhľadania dokumentov v indexoch podľa požiadavky hlavného vyhodnocovacieho systému je na obr. 20.



Obr. 20. Sekvenčný diagram vyhľadania dokumentov v indexe podľa požiadavky

LuceneModul obsahuje metódu *vyhladajDokPodlaDotazu(string dotaz)*, ktorá vráti pole dokumentov v štruktúre *Dokument* s týmito atribútmi:

```
int dokumentID;
private string nazov;
private string subor;
private string autor;
private string kategoria;
private float relevancia;
```

Tieto dokumenty budú naplnené údajmi, ktoré vráti metóda *Search(Query query)* objektu triedy *IndexSearcher* ako pole inštancií triedy *Hits*.

Každý modul v systéme má špecifické úlohy. Vďaka kooperácii modulov systém vyhodnotí a poskytne používateľovi najlepších nájdených kandidátov (dokumentov) podľa jeho charakteristík a požiadaviek. Napriek tomu, že moduly sú na sebe nezávislé, vyhľadávací modul musí niektorým z nich poskytnúť dokumenty s istými vlastnosťami. Preto budú v module implementované nasledujúce funkcie:

- *getDocIdsFromQuery(String[] keywords)*
  Vrátenie množiny identifikátorov dokumentov, v ktorých sa vyskytujú kľúčové slová.

- *getDocIdsFromQuery(String[] keywords, int maxCount)*
  Vrátenie množiny identifikátorov dokumentov, v ktorých sa vyskytujú kľúčové slová, pričom maximálna veľkosť množiny je daná druhým argumentom.

- *getDocIdsFromQueryWithRelevance(String[] keywords, float relevance)*
  Vrátenie množiny identifikátorov dokumentov, v ktorých sa vyskytujú kľúčové slová s relevanciou vyššou ako je hodnota druhého argumentu.

- *getDocIdsFromQueryWithRelevance(String[] keywords, int maxCount, float relevance)*
  Vrátenie množiny identifikátorov dokumentov, v ktorých sa vyskytujú kľúčové slová s relevanciou vyššou ako je hodnota tretieho argumentu. Maximálna veľkosť množiny je daná druhým argumentom.

Všetky tieto funkcie vracajú pole identifikátorov.

3.4 Modul pre kategórie dokumentu

Reprezentácie matice mier podobnosti medzi kategóriami

Táto matica uchováva reálne čísla - miery podobnosti medzi kategóriami. Čísla môžu nadobúdať hodnoty z intervalu <0.0, 1.0>, pričom ak dve kategórie spája hodnota 1.0, znamená to, že sa kategórie podobajú na 100%. V matici je potrebné mať zadefinované všetky kategórie dokumentov. Tieto kategórie, ako aj potrebné hodnoty, je nutné zadať manuálne, napríklad do konfiguračného súboru, odkiaľ budú do matice načítané. Keďže je matica symetrická, stačí zadať iba jednostranné prepojenie. Čo sa týka kategórií netreba zabúdať aj na možnosť, že súbor do žiadnej z nich spadať nemusí. Vtedy treba do konfiguračného súboru napísať novú kategóriu a hodnoty, udávajúce prepojenia s ostatnými, alebo mu prideliť kategóriu „nešpecifikovaná kategória“.

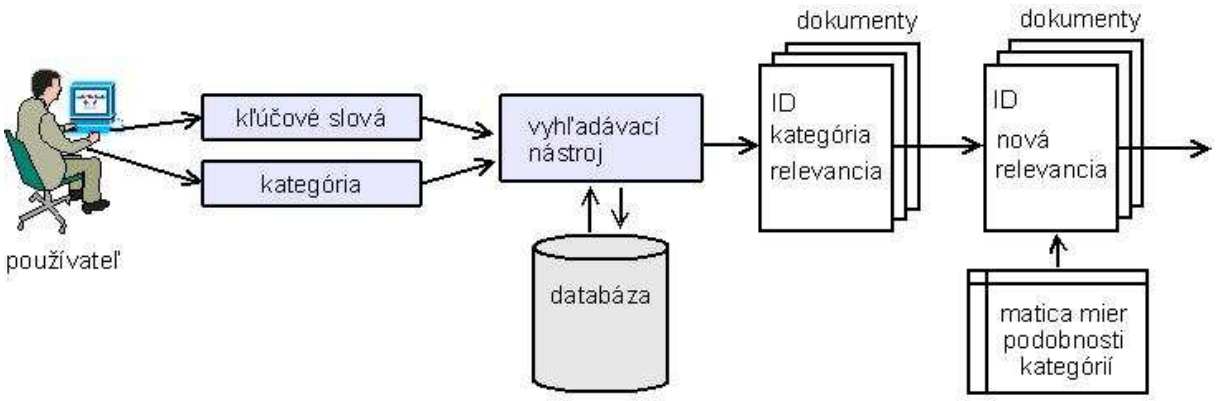
Kategorizácia dokumentov

Kategóriu každého dokumentu je potrebné špecifikovať už pri jeho vkladaní do databázy, pričom kategória, ktorú administrátor priradí dokumentu, sa musí nachádzať v už spomínanej matici mier podobnosti. Následne dokument putuje do databázy, kde sa mu v tabuľke „Dokument“ do atribútu „Kategoria“ zapíše kategória.

Vyhľadávanie dokumentov

Postupnosť krokov vyhľadávania dokumentov začína u používateľa, ktorý zadá kľúčové slová a špecifikuje kategóriu dokumentov, alebo sa kategória bude považovať za bližšie nešpecifikovanú. Systém na

vyhľadávanie dokumentov potom podľa kľúčových slov vyhľadá skupinu dokumentov a vráti ich ID spolu s relevanciou a kategóriou, do ktorej dokument patrí. Kategória, ktorú používateľ zadal sa následne porovná podľa matice mier podobnosti s kategóriami dokumentov, ktoré vyhľadávací nástroj ponúkol. Miery pre každý dokument sa prenásobia s jeho relevanciu, čím sa získa nová relevancia. Takto ohodnotené dokumenty sa potom poskytujú na ďalšie spracovanie. Celý postup je zachytený na obr. 21.



Obr. 21. Proces vyhľadávania a ohodnocovania dokumentov na základe kategórií

Výpočet relevancie dokumentu na základe kategórie

Ako už bolo spomenuté, modul, ktorý bude mať na starosti výpočet relevancie dokumentov na základe kategórií, dostáva na vstupe dokumenty vyhľadané z databázy vyhľadávacím nástrojom, ktorý im priradí aj určitú relevanciu, podľa výskytu kľúčových slov. Ďalšie dôležité vstupy sú matica mier podobnosti kategórií dokumentov a samotná kategória špecifikovaná používateľom. Na základe kategórií dokumentov vyhľadaných v databáze a kategórie špecifikovanej používateľom sa z matice priradia jednotlivým dokumentom miery podobnosti. Každý dokument bude mať teda priradenú mieru podobnosti a relevanciu. Tieto dve čísla sa medzi sebou vynásobia, čím vzniknú nové relevancie priradené dokumentom. Tieto dokumenty, ako aj ich relevancie sa potom ďalej spracovávajú.

Príklad výpočtu relevancie:

Vstupy:

Kategória zadaná používateľom:
Zdroj. kód C#

Matica mier podobnosti medzi kategóriami:

	ZKC#	*ZKJava*	*PP*	*TD*	*ND*
ZKC#	1.00	0.75	0.10	0.30	0.50
ZKJava	0.75	1.00	0.10	0.30	0.50
PP	0.10	0.10	1.00	0.30	0.50
TD	0.30	0.30	0.30	1.00	0.50
ND	0.50	0.50	0.50	0.50	0.50

Legenda:
ZKC# – Zdroj. kód C#
ZKJava – Zdroj. kód Java
PP – Používateľská príručka
TD – Technická dokumentácia
ND – Nešpecifikovaný dokument

Dokumenty:
Dokument A {ID = 1, kategória: *ZKC#*, relevancia: 0.6}
Dokument B {ID = 3, kategória: *ZKJava*, relevancia: 0.5}
Dokument C {ID = 4, kategória: *PP*, relevancia: 0.4}
Dokument D {ID = 7, kategória: *ND*, relevancia: 0.3}
Dokument E {ID = 9, kategória: *TD*, relevancia: 0.2}

Výpočet:

Dokument A: 0.6 * 1.0 = 0.6
Dokument B: 0.5 * 0.75 = 0.375
Dokument C: 0.4 * 0.1 = 0.04
Dokument D: 0.3 * 0.5 = 0.15
Dokument E: 0.2 * 0.3 = 0.06

Výstup:

1.) Dokument A {ID = 1, relevancia: 0.6}
2.) Dokument B {ID = 3, relevancia: 0.375}
3.) Dokument D {ID = 7, relevancia: 0.15}
4.) Dokument E {ID = 9, relevancia: 0.06}
5.) Dokument C {ID = 4, relevancia: 0.04}

3.5 Modul pre grafové algoritmy

Úlohou tohto modulu je prehľadávanie stavového priestoru tvoreného dokumentmi a prepojeniami medzi nimi. Modul sa snaží nájsť dokumenty relevantné ku kľúčovým slovám zadaným používateľom. Na určenie relevantnosti jednotlivých dokumentov používa algoritmus NodeRank.

Vstupy Modulu

Modul dostane na vstupe zoznam kľúčových slov ktoré boli zadané používateľom.
Maximálny počet relevantných dokumentov, ktoré budú na výstupe modulu.

Výstup modulu

Zoznam relevantných dokumentov k zadaným kľúčovým slovám.

Komunikácia s okolím

V rámci systému modul komunikuje s Vyhľadávacím modulom, ktorý mu poskytuje zoznam dokumentov ktoré obsahujú najväčší počet kľúčových slov zadaných používateľom. V databáze má modul uložené prepojenie medzi jednotlivými dokumentmi , to znamená ID jednotlivých dokumentov a taktiež typ prepojenie ktorý je medzi nimi. Modul poskytuje svoje výstupy Rozhodovaciemu modulu.

Požiadavky používateľa sú modulu zadávané prostredníctvom grafického rozhrania priamo používateľom.

Činnosť modulu

Obr. 22. Graf činnosti modulu

Činnosť modulu spočíva v určení zoznamu dokumentov ktoré sú relevantné ku kľúčovým slovám.

Prvým krokom je získanie kľúčových slov, ktoré zadáva priamo používateľ. Po získaní týchto kľúčových slov modul pošle požiadavku na získanie ID dokumentu ktorý má najväčší počet podobných kľúčových slov ako v zadanom zozname. Toto ID získa od Vyhľadávacieho modulu.

Na základe tohto ID module prehľadá graf tvorený dokumentmi a prepojeniami medzi nimi a určí dokumenty ktoré sú s ním relevantné. Na určenie relevancie sa použije algoritmus NodeRank, ktorý je bližšie popísaný v časti analýza.

Vytváranie grafu

Pri vytváraní grafu z dokumentov a prepojení medzi nimi vychádzame z predpokladu, že pokiaľ je dokument A prepojený z dokumentom B je aj dokument B prepojený z dokumentom A. Po pridaní nového dokumentu do systému sa tento automaticky indexuje za použitia vyhľadávacieho modulu.

Následne sa v ňom určia priame odkazy na iné dokumenty a ich názvy. To znamená, že pokiaľ nový dokument obsahuje názov iného dokumentu, prípadne cestu na diskovú jednotku s týmto súborom identifikuje sa prepojenie medzi týmito dvoma dokumentmi.

Po vytvorení priamych prepojení dôjde k vytvoreniu prepojení na základe kľúčových slov v kapitolách, názvoch a celých dokumentov. Pre vytvorenie týchto prepojení bude potrebná spolupráca s Vyhľadávacím modulom. Vyhľadávací modul poskytne zoznam dokumentov, ktoré sa na určitý počet percent podobajú na nový dokument. To znamená že obsahujú určitý počet podobných slov. Pre začiatok bude použitá hodnota 80%, ktorá sa môže v procese implementácie zmeniť. Následne sa zaznamenajú nájdené prepojenia do databázy.

Obr. 23. Vytváranie prepojení dokumentu

3.6 Modul pre sociálne siete

Pri analýze tejto problematiky sa hovorilo o dvoch súčastiach vytvárania používateľského modulu: explicitné a implicitné. V tejto kapitole sa uvádza hrubý návrh týchto dvoch komponentov.

Manuálny komponent, slúžiaci na vytváranie profilov bude mať tieto hlavné funkcie:

- Vytvorenie nového profilu
- Zmena údajov v danom profile
- Prípadne vymazanie konta(najlepšie ak túto funkciu priradíme administrátorovi)

Editovanie profilu bude možné po prihlásení sa do systému, pričom prvé prihlásenie sa do systému je podmienené vytvorením konta. Tieto funkcie sú už všeobecne známe z webových portálov, stačí si napríklad zobrať vytváranie konta pre e-mail. Dôraz pri návrhu kladieme na navrhnutie údajov, ktoré vytvárajú profil používateľa. Navrhujeme tieto položky:

- prihlasovacie meno
- prihlasovacie heslo
- osobné údaje(meno, titul, kontakt)
- oddelenie, v ktorom používateľ pracuje(zadefinované administrátorom)
- témy, ktoré požívateľa zaujímajú
- oblasti v ktorých má dobré skúsenosti

V profile sa zachovajú aj údaje na akom aktuálnom projekte používateľ pracuje. A budú sa pamätať aj údaje, ktoré dokumenty pridal do systému. To sa však viacej týka explicitného komponentu, ktorý zaznamenáva interakciu používateľa so systémom.

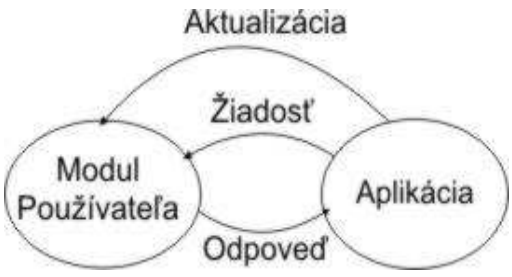
Dynamický komponent umožní sledovať tieto akcie systému:

- Pridávanie dokumentov do bázy dát
- Vyhľadávanie dokumentov, zadávanie parametrov vyhľadávania
- Vyberanie dokumentov z alternatívne poskytnutých

Údaje(získané z interakcie), ktoré sa uložia do modelu používateľa:

- Dokumenty, ktoré používateľ pridal do databázy
- Dokumenty, vytvorené autorom
- Ktorý dokument koľkokrát vyhľadával
- Počet využitých alternatív, tým sa dá overiť nakoľko spĺňa systém používateľove požiadavky a meniť údaje na základe týchto dát

Samotná komunikácia aplikácie s týmto modulom bude založená na posielaní správ a získavaní odpovedí. Z aplikácie sa pošle žiadosť na údaje o používateľovi, tieto sa zahrnú do vyhľadávania dokumentov. Po vyhľadaní sa aktualizuje profil používateľa na základe jeho správania sa. Znázornenie tejto myšlienky je zobrazené na obr. 24. [5]



Obr. 24. Komunikácia modulu s aplikáciou

3.7 Modul modelovania používateľov

Doteraz navrhovaný komponent modelovania používateľov sme mierne zmenili oproti zámerom, ktoré sme plánovali prvý semester tímového projektu. Vtedy išlo skôr o modelovanie používateľov pomocou ontológií. Samotná myšlienka by vyhovovala, keby náš systém bol rozsiahlejší a potrebovali by sme voľnejšiu ruku pri nastavení parametrov aplikácie. Rozhodli sme sa implementovať pôvodne plánovaný modul podobným spôsobom ako fungujú návrhové systémy. Týmto ťahom sme vlastne zachovali myšlienku rozdelenia komponentu na manuálnu časť - pomocou ktorého sa vytvárajú profily a na dynamickú časť, ktorá sa mala prispôsobiť používateľovi. Vytváranie profilu má svoje výhody aj pre ďalšie komponenty aplikácie. Je to uvedené pri architektúre systému, že vlastne ide o klient-server architektúru a jednotlivé moduly sú od seba nezávislé, používateľ si vyberá na základe uváženia, ktoré moduly sa majú použiť pri vyhľadávaní. Z tohto hľadiska je dôležité vedieť ako sa používateľ správa, teda jeho históriu. Najmä treba poznať, ktoré dokumenty otváral a koľkokrát tak daný používateľ učinil. V nasledujúcej časti tejto kapitoly si vysvetlíme základný koncept návrhových systémov.
Návrhové systémy si môžeme predstaviť ako určité typy dátových filtrov [14], ktoré zobrazujú používateľom aktuálne informácie o predmetoch na ktoré podali dotaz alebo o ktoré sa zaujímajú. Znalostný manažment disponuje rozsiahlou bázou znalostí, ktorá predstavuje vlastne množinu dokumentov. Z toho vyplýva, že náš návrhový systém bude postavený na návrhu dokumentov. Otázkou zostáva ako nazbierať potrebné informácie k poskytnutiu návrhov. Tu treba spomenúť techniku kolaboratívneho filtrovania.
Kolaboratívna filtrácia je prístup, ktorá využíva rôzne vzťahy vznikajúce medzi entitami danej aplikačnej domény. Môžu to byť vzťahy medzi agentmi (používatelia), rôznymi pohľadmi na objekty alebo súvislosti zdrojov dát. Aplikácie, kde sa používa kolaboratívne filtrovanie dát sa vyznačujú obrovským množstvom údajov, kde klasické vyhľadávanie môže viesť k prebytočnému počtu výsledkov. Nájdenie správnych a relevantných informácií požaduje inteligentné prehľadanie priestoru. Používa sa tu idea odhadnutia a ohodnotenia typu a "vkusu" agentov a objavenie súvislostí, ktoré sme spomínali. Celá funkcionalita sa dá popísať dvoma krokmi:

1. Nájdu sa používatelia s podobným ohodnotením ako má aktuálny používateľ, teda pre ktorého sa návrh uskutočňuje.
2. Použijú sa dostupné informácie od používateľov nájdených v prvom kroku na zostavenie predikcie.

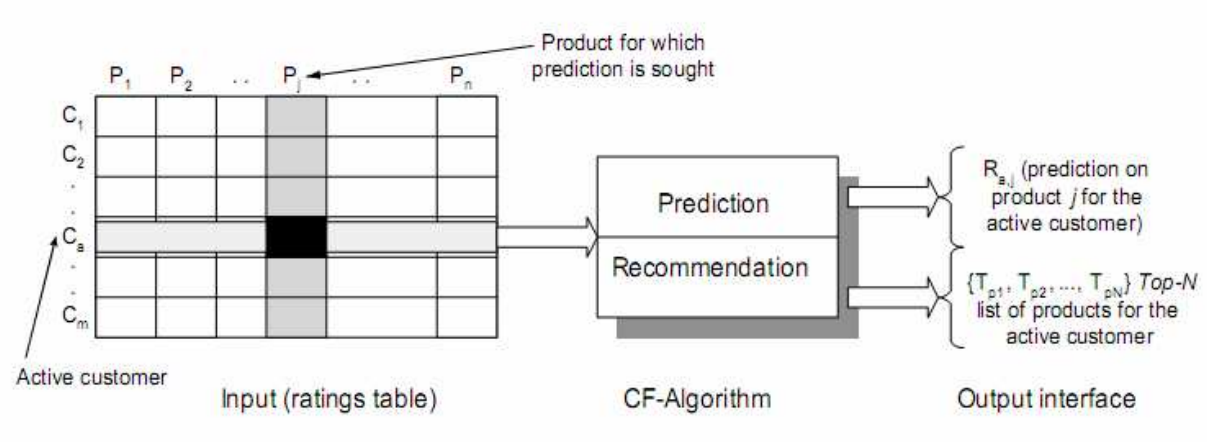
Takýto princíp používa napríklad aj e-bay, kedy sa pri kúpe položky *x* navrhnú iné tovary *y*, ktoré kúpili ostatní kupujúci popri kúpení *x* (používatelia, ktorí kúpili *x* kúpili aj *y*).

Existujú dva rôzne prístupy pre získanie návrhov, jednou je vytvorenie matice podobnosti medzi položkami (entity príslušnej domény – u nás dokumenty), a druhým je tvorba matice podobnosti (susednosti) používateľov. Oba prístupy sú podobné, ide vlastne len o rôzny pohľad na situáciu. Centrálnou záležitosťou je hodnotenie položiek používateľmi. Podľa spôsobu hodnotenia rozlišujeme aktívnu a pasívnu filtráciu.

Aktívne filtrovanie používa peer-to-peer prístup [14], t.j. sú to systémy, kde používatelia hodnotia dané položky, tovar, produkty v systéme. Takéto hodnotenia môžu aj zdielať, teda sprístupniť pre ostatných. Je to veľmi výhodné, keď sa o neznámom alebo málo známom predmete dozvieme pripomienky a hodnotenia ostatných. Ak je zabezpečená aj vierohodnosť informácií, nadobudnuté znalosti sú naozaj užitočné. Najväčším rozdielom medzi aktívnou filtráciou a filtráciou na základe obsahu je, že pri aktívnej filtrácii sa vôbec nepočíta s ozajstnou hodnotou položiek. Môžeme poskytnúť návrhy bez toho, aby sme presne poznali obsah ponúkaných predmetov. Aktívna selekcia má svoju nevýhodu pri akciách ako sú nové položky v systéme alebo registrácia nového používateľa. Bez ohodnotenia nemôže byť položka navrhovaná, naopak u novom používateľovi ešte nepoznáme jeho preferencie, musí najprv ohodnotiť niektoré predmety. Problémy vznikajúce týmito akciami sa nazývajú problém prvého hodnotenia (first rater) a problém studeného štartu (cold start).

Pasívne filtrovanie sa vyznačuje zbieraním údajov o akciách vykonaných v systéme [14], je to vlastne implicitné získavanie dát. Sledujeme napríklad aké predmety daný používateľ kúpil, koľkokrát pozrel niečo a sledujeme súvislosti.

Existuje ešte aj takzvané filtrovanie položiek, ktoré vytvára určité skupiny predmetov, na základe hodnotení používateľov. Potom vychádzajúc z reakcií používateľov sa podobne vytvoria skupiny používateľov. Výhodu to má v zavedení nového stupňa abstrakcie a zmenšíme tým rozsah prehľadávaného priestoru.
Proces kolaboratívneho filtrovania je uvedený na obrázku č. 25.



Obr. 25. Proces kolaboratívneho filtrovania [15]

3.8 Webová služba poskytujúca funkcie znalostného manažmentu

Služba bude umiestnená na serveri. Bude to trieda *FoobeServis* rozširujúca triedu *System.Web.Services.WebService*. Poskytne klientským aplikáciám metódy pre narábanie so znalostným manažmentom a s kontami používateľov.

Metódy pre spravovanie používateľov:
pridajPouzivatela(string, string)– pridanie používateľa do databázy (prihlasovacie meno a heslo)
odstranPouzivatela(string) – odstránenie používateľa z databázy (prihlasovacie meno)
overLogin(string, string) – overenie prihlasovacieho mena a hesla používateľa pri prihlasovaní

Metódy pre nastavenie znalostného manažmentu:
zmenNastaveniaDatabazy(string, string, string, string) – nastavenie pripojenia na databázu
vratNastaveniaDB() – vráti nastavenia pripojenia na databázu
testServisu() – otestovanie funkčnosti servisu a vrátenie jeho nastavení

Metódy pre spravovanie bázy znalostí:
pridajDokument(string, string, string, string, byte[], byte[]) – pridanie dokumentu do bázy znalostí (názov dokumentu, autor dokumentu, názov súboru, kategória dokumentu, binárna forma súboru, kontrolná suma súboru)
odstranDokId(int id) – odstránenie dokumentu s príslušným identifikátorom
vyhladajDokumenty(string, string, string, string) – vyhľadanie dokumentov podľa požiadaviek používateľa (argumenty sú vyhradené pre dotaz pre vyhľadávací modul, kategóriu, prihlasovacie meno používateľa a použité moduly pre vyhľadávanie v báze znalostí)
stiahniDokument(string, id) – odoslanie dokumentu so zadaným identifikátorom používateľovi s prihlasovacím menom uvedeným v prvom argumente

Činnosť webovej služby bude zaznamenávaná do protokolového súboru.

3.9 Aplikácie na strane klienta

Aplikácie pre komunikáciu so systémom umiestneným na serveri budú poskytovať grafické používateľské rozhranie. Budú implementované dva typy aplikácií:

- administratívne
- vyhľadávanie

Administratívne grafické používateľské rozhranie
Umožní volať metódy webovej služby *FoobeServis* pre pridanie používateľa, odstránenie používateľa, pridanie dokumentu, odstránenie dokumentu a nastavenie pripojenia k databáze.

Vyhľadávanie
Ponúkne používateľovi rozhranie na vyhľadávanie v báze znalostí. K vyhľadaným dokumentom bude uvedený názov, autor, kategória a relevancia vyhľadania. Používateľ bude mať možnosť stiahnuť označený dokument.

Rozhranie bude doplnené o rozširujúce možnosti ďalšími oknami:

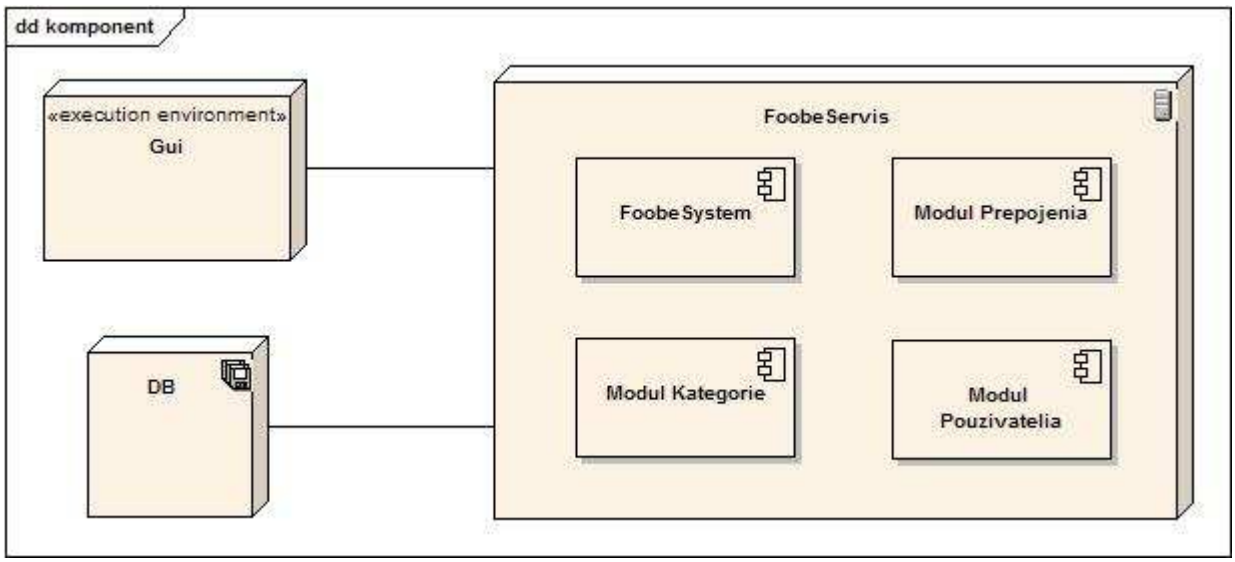
- nastavenie servera, na ktorom beží systém znalostného manažmentu
- nastavenie váh modulov používaných pri vyhľadávaní v báze znalostí
- nastavenie váh kategórií

4 Technická dokumentácia

Architektonicky sa systém skladá z troch častí.

- aplikácie na strane klienta
- webová služba so systémom pre znalostný manažment
- databáza

Ich činnosť z pohľadu spolupráce je znázornená na obr. 26.



Obr. 26. Komponenty systému *Foobe*

Klientské aplikácie poskytujú rozhrania, ktoré volajú metódy webovej služby *FoobeServis*. Tá spolupracuje s modulmi a databázou, čím vzniká priestor pre znalostný manažment.

Základom je *FoobeSystem*, ktorý implementuje vyhľadávací modul *LuceneModul*. Ten poskytuje funkcie pridávania a odstraňovania dokumentov, ako aj vyhľadávanie informácií v indexe. Ostatné moduly sú používané webovou službou pri vyhľadávaní dokumentov.

Pri odoslaní požiadavky z aplikácie služba eviduje, ktoré moduly sa majú pri vyhľadávaní aktivovať. Následne každý modul poskytne výsledky, a tie sa potom spracujú v hlavnom výpočtovom procese pre získanie relevancií dokumentov.

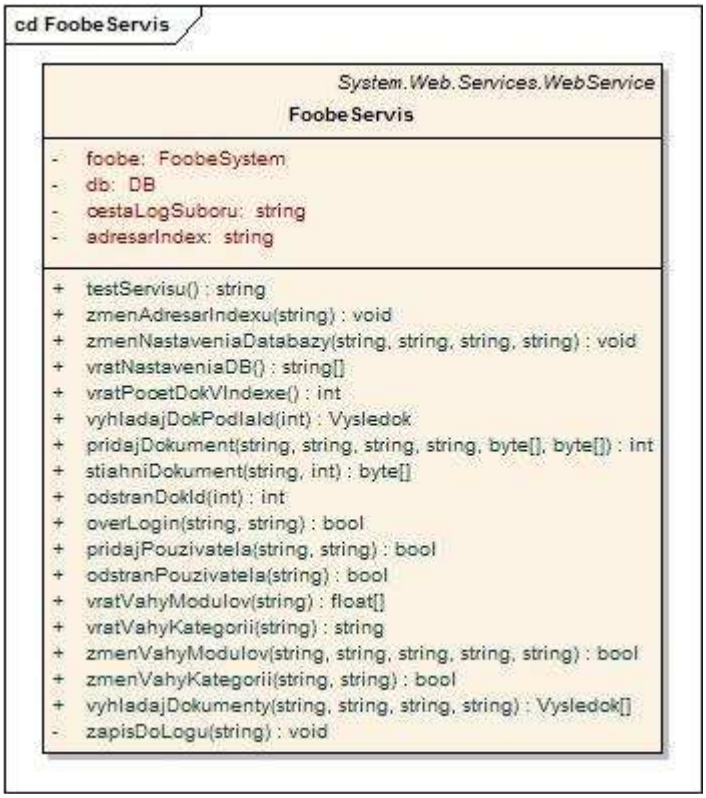
Používateľ má možnosť nastaviť váhy modulov, čím aktívne ovplyvňuje a podporuje vyhľadávanie dokumentov podľa jeho požiadaviek.

Nasledujúce kapitoly obsahujú popis jednotlivých komponentov a modulov.

4.1 FoobeServis

Trieda poskytujúca znalostný manažment v architektúre klient – server. Je spustená na serveri IIS, na ktorom je umiestnený index vyhľadávacieho modulu. Prepojenie s databázou zabezpečuje prístup k dokumentom a informáciám, ktoré systém uchováva (informácie o používateľoch, dokumenty evidované znalostným manažmentom, dáta modulov).

Diagram pre túto triedu sa nachádza na obr. 27.



Obr. 27. Diagram triedy FoobeServis

Atribúty:
foobe – základná súčasť systému obsahujúca vyhľadávací modul
db – inštancia triedy DB, ktorá poskytuje funkcie na prácu s databázou
cestaLogSuboru – cesta k súboru, do ktorého sa zapisuje činnosť služby
adresarIndex – adresár na serveri, v ktorom sa nachádza index vyhľadávacieho modulu

Metódy:
testServisu() – otestovanie, či je servis funkčný a základné nastavenia (adresár služby, adresár indexu, databáza)
zmenAdresarIndexu(string) – nastavenie cesty k indexu vyhľadávacieho modulu
zmenNastaveniaDatabazy(string, string, string, string) – nastavenie pripojenia k databáze
vratNastaveniaDB() – vráti nastavenia pripojenia k databáze
vratPocetDokVindexe() – vráti počet dokumentov v indexe v adresári indexu (atribút *adresarIndex*)
vyhladajDokPodlaId(int) – vyhľadanie dokumentu s príslušným identifikátorom
pridajDokument(string, string, string, string, byte[], byte[]) – pridanie dokumentu do bázy znalostí
stiahniDokument(string,int)– vráti pole byte-ov, ktoré predstavujú binárnu formu súboru, ako reťazec medzi argumenty vstupuje prihlasovacie meno používateľa, ktorý súbor sťahuje (pre účely modulov)
odstranDokId(int) – odstránenie dokumentu z bázy znalostí
overLogin(string, string) – overenie prihlasovacieho mena a hesla používateľa
pridajPouzivatela(string, string) – pridanie nového používateľa (prihlasovacie meno a heslo)
odstranPouzivatela(string) – odstránenie používateľa (prihlasovacie meno)
vratVahyModulov(string) – vráti pole typu float, ktoré predstavuje nastavenia preferenčných váh pre jednotlivé moduly používateľa daného v argumente
vratVahyKategorii(string) – vráti reťazec, ktorý používa modul *Kategorie* na určenie vzdialeností kategórií dokumentov podľa preferencií daného používateľa

zmenVahyModulov(string, string, string, string, string) – zmena preferenčných váh modulov, na prvom mieste je prihlasovacie meno používateľa, ktorý tieto váhy mení

zmenVahyKategorii(string, string) – nastavenie vzdialeností kategórií pre daného používateľa (1. argument)

zapisDoLogu(string) – zapíše text do protokolového súboru umiestneného v *cestaLogSuboru*

vyhladajDokumenty(string, string, string, string) – vyhľadanie dokumentov v báze znalostí

1. argument – dotaz pre vyhľadávací modul
2. argument – kategória vyhľadávania
3. argument – prihlasovacie meno používateľa
4. argument – aktívne moduly v reťazci ako znak '1' (vyhľadávací modul, modul kategórií, modul prepojení, modul podobných používateľov)

návratová hodnota – *Vysledok[]* (štruktúra popisujúca dokumenty, ktoré sú výsledkom vyhľadávania) – viď. obr. 28.

```
public struct Vysledok
{
    public int id;
    public string nazov;
    public string autor;
    public string subor;
    public string kategoria;
    public float relevancia;
    public string sposobVyhladania;
    public string ukazka;
}
```

Obr. 28. Štruktúra na vrátenie výsledkov vyhľadaných dokumentov

Proces prideľovania relevancií dokumentom je v tejto metóde vykonávaný postupne. Pre každý modul sa vyhodnotí vyhľadávanie v báze znalostí a nájdené dokumenty sa spolu s relevanciami v danom module zapíšu do poľa štruktúry *VysledokModulu*, ktorý obsahuje dva atribúty. Prvým je identifikátor dokumentu (int) a druhým relevancia (float).

Na záver sa dokumenty uložia do jedného poľa, pričom každému dokumentu sa priradí relevancia nasledovne:

$$\sum \text{váha aktívneho modulu x relevancia dokumentu pre daný modul}$$

Činnosť služby je zaznamenávaná do protokolového súboru, ktorého umiestnenie je v atribúte *cestaLogSuboru*. Ku každej operácii je evidovaný presný čas jej vykonania, ip adresa klienta, z ktorého bola požiadavka odoslaná a popis operácie s výsledkami. Ukážka protokolového súboru je na obr. 29.

```
12. 5. 2008 13:41:16 147.175.182.42 : testServisu -> servis Foobe je funkcny
server: PAXO
bazovy adresar: C:\Inetpub\wwwroot\FoobeServis
index je v adresari "index"
databaza: server -> paxo, nazov -> foobeDB ... spojenie funkcne
12. 5. 2008 17:41:18 147.175.182.47 : overLogin (jano) -> True
12. 5. 2008 19:45:36 147.175.182.47 : vyhladajDokumenty (a*, Analýza, jano, 1100) -> najdenych 9 dokumentov
12. 5. 2008 20:04:21 147.175.182.47 : stiahniDokument(jano, 1036)
12. 5. 2008 20:04:54 147.175.182.42 : odstranPouzivatela (dano) -> True
```

Obr. 29. Ukážka z protokolového súboru

4.2 Databázový modul

Úlohou databázového modulu je poskytovať rozhranie pre prácu s databázou ostatným modulom programu. Oddelenie databázovej časti systému od samotnej logiky programu bolo realizované z dôvodu zjednodušenia práce na programe. Išlo predovšetkým o odľahčenie práce programátorov na ostatných moduloch, ktorý sa nemuseli zaoberať prácou s databázou a nemuseli riešiť problémy vzniknuté pri práci s databázou.

Databázový modul poskytuje funkcie pre nasledovné moduly:

- Modul pre kategórie dokumentu
- Modul pre prepojenia dokumentov
- Module pre používateľov

Databázový modul pracuje nad databázovým serverom Microsoft SQL server 2005. V nasledujúcej časti budú popísané jednotlivé funkcie poskytované databázovým modulom.

Vytvorenie modulu a pripojenie sa k databázovému serveru

Pre začatie práce s databázovým modulom je potrebné najskôr vytvoriť objekt triedy *DB,* ktorý poskytuje funkcie databázového modulu. Vytvorenie objektu prebieha za pomoci konštruktora *DB()*, ktorému sa nepredávajú žiadne parametre. Novovvytvorený objektu typu *DB* nieje zatiaľ schopný žiadnej činnosti. Pre začatie činnosti modulu je potrebné pripojiť ho k databázovému serveru.

Pripojenie na databázový server prebieha pomocou zavolania procedúry

public void pripojSa(String server, String databaza, String login, String heslo)

Ktorá dostáva nasledovné parametre:

- **server** – Názov databázového servera (napr: localhost\\sqlexpress)
- **databaza** – Názov databázy na databázovom servery
- **login** – Prihlasovacei meno do databázy
- **heslo** – Heslo prislúchajúce k prihlasovaciemu menu

V prípade neúspešného pripojenia na server vznikne výnimka a činnosť modulu bude ukončená.

Funkcie pre prácu so súbormi

Tieto funkcie v databázovom modely poskytujú rozhranie pre pridávanie a vyberanie súborov v databáze. Pridávanie súboru je realizované pri pridaní nového súboru do systému. Vyberanie súboru prebieha pri vyhľadávaní a zobrazovaní súborov požívateľom.

Procedúra

public void vlozSubor(int ID, String nazov, String autor, int kategoria, String subor)

poskytuje rozhranie pre pridanie nového súboru do databázy jej parametre sú nasledovné:

- **ID** - Identifikačné číslo, ktoré chceme aby bolo priradené súboru. Toto číslo je vygenerované počas procesu indexácie súboru systémom .Lucene. Toto číslo je pre každý súbor v systéme jedinečné, preto je ho možné použiť ako jednoznačný identifikátor súboru.
- **nazov** - Názov ktorý chceme priradiť súboru.
- **autor** - Meno autora alebo autorov dokumentu, v prípade že sa jedná o viacero autorov sú ich mená oddelené bodkočiarkov.
- **kategori**a - Identifikačné číslo kategórie do ktorej dokument patrí.
- **subor** - cesta k súboru, ktorý má byť vložený do systému

Funkcia

public Zaznam vyberSubor(int ID)

Zabezpečuje výber dokumentu z databázy na základe je ID. Ako návratovú hodnoty vracia objekt typu *Zaznam*, ktorý obsahuje všetky informácie o súbore s daným ID uložené v databáze. Objekt Zaznam obsahuej nasledujúce atribúty:

int ID;
String nazov;
String autor;
int kategoria;
byte[] subor;

Význam jednotlivých atribútov je zrejmý z ich názvu. Atribút *subor* obsahuje pole bajtov, ktoré reprezentujú uložený súbor.

public bool existujeDokument(int ID)

Pomocná funkcia, ktorej úlohou je zistiť či sa dokument s daným ID nachádza v databáze. Ako vstupný argument funkcia dostáva identifikačné číslo dokumentu. Návratovou hodnoutou funkcie je *true* ak sa dokument v databáze nachádza a *false* ak sa dokument v databáce nenechádza.

Funkcie pre prácu s používateľmi

Nasledujúce funkcie sú využívané modulom pre používateľov. Poskytujú rozhranie pre prácu tohto modulu s databázou.

public bool pridajPouzivatela(String login, String heslo)

Pridá nového používateľa do databázy. V prípade úspechu vráti hodnotu *true*, v prípade neúspechu hodnotu *false*. Hodnotu false vracia funckia aj v prípade ak v databáze už existuje používateľ s rovnakým loginom ako je login požadovaný pri volaní funkcie.
Parametre funkcie sú nasledovné:

- login - Login nového používateľa
- heslo - Heslo pre nového používateľa

pridajPouzivatela(String login, String heslo, double vaha1, double vaha2, double vaha3, double vaha4)

Funkcia má rovnakú úlohu aj návratovú hodnotu ako *pridajPouzivatela(String login, String heslo)*, rozdielom je počet parametrov funkcie. V tomto prípade sa k danému používteľovi zapisujú aj váhy, ktoré prikladá jednotlivýv modulom.

public void zmenVahy(int ID_pouzivatela, double nova_vaha1, double nova_vaha2, double nova_vaha3, double nova_vaha4)

Táto procedúra slúži pre nastavenie jednotlivých váh modulom v závislostí od preferencií používateľa. Na vstupe dostáva nasledovné atribúty:

- ID_pouzivatela - identifikačné číslo používateľa, ktorého váhy sa idú meniť
- nova_vaha1 – nova_vaha4 - váhy prislúchajúce jednotlivým modulom

public Pouzivatel vratPouzivatelaPodlaID(int ID_pouzivatela)

Funkcia, ktorá vracia informácie o používateľovi z databázy na základe jeho identifikačného čísla. Ako parameter dostáva identifikačné číslo používateľa. V prípade úspechu vracia objekt typu používateľ, popisujúci daného používateľa. Ak sa daný používateľ v databáze nevyskytuje vráti hodnotu *null*.

Trieda používateľ obsahuje nasledovné atribúty:

- **int ID;**
- **String login**
- **String heslo**
- **float vaha1**
- **float vaha2**
- **float vaha3**
- **float vaha4**

public Pouzivatel vratPouzivatelaPodlaLoginu(String login_pouzivatela)

Podobná funkcia ako *vratPouzivatelaPodlaID(int ID_pouzivatela)*, návratové hodnoty sú rovnaké, jediný rozdiel je v tom, že používateľa vracia na základe jeho loginu, ktorý dostáva ako jediný parameter.

public bool skontrolujLogin(String login, String zadane_heslo)

Úlohou tejto funkcie je skontrolovať správnosť hesla, ktoré bolo zadané spolu s loginom. Login a heslo dostáva ako svoje dva parametre. Funkcia porovná heslo, ktoré prislúcha k danému loginu v databáze so zadaným heslom. V prípade že sa heslá zhodujú vráti hodnotu *true*. Inak vráti hodnotu *false*. Hodnotu *false* vráti aj v prípade ak používateľ s daným loginmi v databáze neexistuje.

public void pridajDokumetnPouzivatelovi(int dokument, int pouzivatel)

Procedúra, ktorá vloží do databázy záznam o tom, že používateľ otvoril určitý súbor a pracoval s ním. Argumentami funkcie sú identifikačné číslo dokumentu a identifikačné číslo používateľa ktorý s daným dokumentom pracoval.

Funkcie pre prácu s prepojeniami

Tieto funkcie sú využívané modulom pre prepojenia medzi dokumentami. Ich úlohou je poskytovať rozhranie pre pridávanie vyhľadávanie prepojení medzi dokumentami v databáze.

public Prepojenie[] vratPrepojenia(int ID_dokumentu)

Funkcia, ktorá vráti všetky prepojenia medzi dvoma dokumentami uložené v databáze. Vstupnými argumentami identifikačné číslo dokumentu ku ktorému hľadáme prepojenia. Výstupnou hodnotou je pole objektov typu prepojenie.

Objekt prepojenie prepojenie reprezentuje prepojenie medzi dvoma dokumentami. Objekt prepojenie obsahuje nasledujúce atribúty.

- **int dokument_ID – Identifikačné číslo dokumentu, ktorý je prepojený s dokumentom**
- **int typ - typ prepojenie, určuje či sa jedná o prepojenie na základe autore, názvu dokumentu alebo názvu súboru**
- **float relevancia - relevancia prepojenia medzi dvoma dokumentami**

public void pridajPrepojenie(int dokument1, int dokument2, int typ_prepojenia, float relevancia)

Procedúra, ktorá pridá prepojenie medzi dvoma dokumentami do databázy. Jej vstupné atribúty sú:

- **dokument1** – prvý z dokumentov, medzi ktorými existuje prepojenie
- **dokument2** – druhý z dokumentov, medzi ktorými existuje prepojenie
- **typ_prepojenia** – typ prepojenia(na základe autora, názvu dokumentu alebo názvu súboru)
- **relevancia** - relevancia prepojenia medzi dokumentami

Modul pre prepojenia

Úlohou tohto modulu je poskytovať funkcie pre vyhľadávanie prepojení medzi dokumentami. Modul spolupracuje s nasledovnými modulmi:

- Databázový modul – Tento modul je využívaný na ukladanie nájdených prepojení do databázy. Taktiež sa využíva aj na nájdenie už existujúcich prepojení medzi dokumentami, ktoré sú uložené v databáze.
- Modul .Lucene – Úlohou tohto modulu je z pohľadu modulu pre prepojenia, vyhľadávať dokumenty, obsahujúce reťazce určené ako kľúčové slová.

Prepojenia dokumentov

Prepojenie dokumentov v kontexte programu znamená dvojicu dokumentov, ktorá je určitým spôsobom vzájomne prepojená. Jedná sa o reflexívnu reláciu medzi dvoma dokumentami A, B. Pričom ak je prepojený dokument A z dokumentom B je automaticky prepojený aj dokument B z dokumentom A.

Ku každému prepojenie je priradená aj relevancia prepojenia, ktorá sa vypočítava na základe dvoch faktorov. Prvým z nich je váha typu prepojenia. Každý typ prepojenia je ohodnotení odlišnou váhou. Táto váha je nastavená ako empirická laboratórna konštanta, reprezentujúca pravdepodobnosť, že dva dokumenty sú navzájom relevantné. Viacej o typoch prepojení a ich váhach je popísané v nasledujúcej časti. Druhým faktorom je vzájomná relevancia dvoch dokumentov na základe príslušného reťazca. Táto relevancia sa získa z modulu .Lucene. Výsledná relevancia vznikne ako súčin týchto dvoch veličín.

Medzi dvoma dokumentami môže existovať viacero prepojení. Pokiaľ ide o viacnásobné prepojenie toho istého typu, modul ich považuje a do databázy uloží ako jedno prepojenie s hodnotou relevancie zvýšenou úmerne počtu takýchto prepojení. Ak sa medzi dvoma dokumentami vyskytujú prepojenia rozdielneho typu, každé s týchto prepojení je zaznamenané. Na základe počtu prepojení a ich relevancie sa vypočíta výsledná relevancia dvoch dokumentov vzhľadom na prepojenia.

Typy prepojení

Modul pre prepojenia dokáže spracovávať tri typy vzájomných prepojení dokumentov. Sú to tieto:

- Prepojenie na základe autora – Toto prepojenie existuje v prípade, ak dokument A obsahuje vo svojom texte meno autora, alebo autorov dokumentu B. V tomto prípade je tu určitá pravdepodobnosť, že tieto dva dokumenty navzájom súvisia. Ak má dokument viacero autorov, hľadajú sa postupne prepojenia pre meno každého z nich. Ako meno autora je braný celý reťazec znakov *„meno priezvisko“*, pričom medzera medzi menom a priezviskom sa neberie ako oddeľovací znak. Problémom pri tomto prístupe môže byť uvádzanie druhé mena alebo prezývky autorov v niektorých dokumentoch. Ak je v jednom dokumente auto uvedený v tvare *„meno druhé_meno priezvisko“* nebude možné ho prepojiť s dokumentom kde je autor uvedený bez druhého mena.
  - Hodnota váhy tohto prepojenia je v programu nastavená na hodnotu 0.5. Hodnota 0.5 bola zvolená z toho dôvodu, že ak sú dva dokumenty vytvorené jedným autorom je tu síce pravdepodobnosť že budú vzájomne relevantné, no na druhej strane jeden autor môže vytvárať dokumenty ktoré sa od seba po obsahovej stránke silne odlišujú.
- Prepojenie na základe názvu dokumentu – Prepojenie na základe názvu dokumentu existuje medzi dokumentami A a B, v prípade ak sa v dokumente A nachádza reťazec, ktorý bol v počas procesu pridávania dokumentu do systému zadaný ako názov dokumentu B. Názov dokumentu môže byť odlišný ako názov súboru, v ktorom je dokument uložený.
  - Váha tohto typu prepojenia je v systéme nastavená na hodnotu 0.9. Hodnota váhy je pomerne vysoká z toho dôvodu, lebo prepojenie na základe názvu dokumentu predstavuje priamy odkaz na iný dokument, a teda tieto dva dokumenty obsahujú obsah k sebe navzájom relevantný.
- Prepojenie na základe názvu súboru – Je podobné ako prepojenie na základe názvu dokumentu, s tým rozdielom že existuje ak dokument A obsahuje názov súboru v ktorom je uložený dokument B.
  - Váha tohto typu prepojenia je nastavená na hodnotu 0.3. Hodnota je pomerne nízka z toho dôvodu, že dva rozdielne dokumenty môžu byť uložené do úborov s rovnakým názvami.

Činnosť modulu

Činnosť modulu možno rozdeliť na dve základné činnosti. Prvou z nich je vyhľadávanie prepojení medzi dokumentami a ich ukladanie do bázy dát. Druhou činnosťou je vyhľadanie dokumentov prepojených z určitým dokumentov na základe práve týchto prepojení.

- Vyhľadávanie prepojení v dokumentoch - Pri pridávaní dokumentu do systému sa automaticky vyhľadajú prepojenie, ktoré spájajú nový dokument s dokumentami už existujúcimi v systéme. Vyhľadávanie prepojení sa realizuje za pomoci procedúry

  najdiPrepojenia(Dokument dokument, Vyhladavac vyhladavac)

  Procedúra dostáva ako svoje parametre objekt typu *Dokument* popisújúci novo pridávaný súbor. A

objekt typu *Vyhladavac*, ktorý umožňuje vyhľadávanie už existujúcich dokumentov. Procedúra najskôr nájde prepojenie dokumentov cez názov dokumentu, následne cez autora a na záver prepojenia na základe názvu súboru. Nájdené prepojenia sú po vypočítaní ich relevancie automaticky ukladané do databázy.

- Vyhľadávanie dokumentov na základe prepojení - Na už nájdenýc dokumentov vyhľadávací modul vráti zoznam dokumentoc a ich relevancií, ktoré boli nájdené na základe prepojení dokumentov. Pre nájdenie dokumentov sa využíva funkcia:

  DokRel[] vratDokumety(Dokument[] zaciatocne_dokumenty

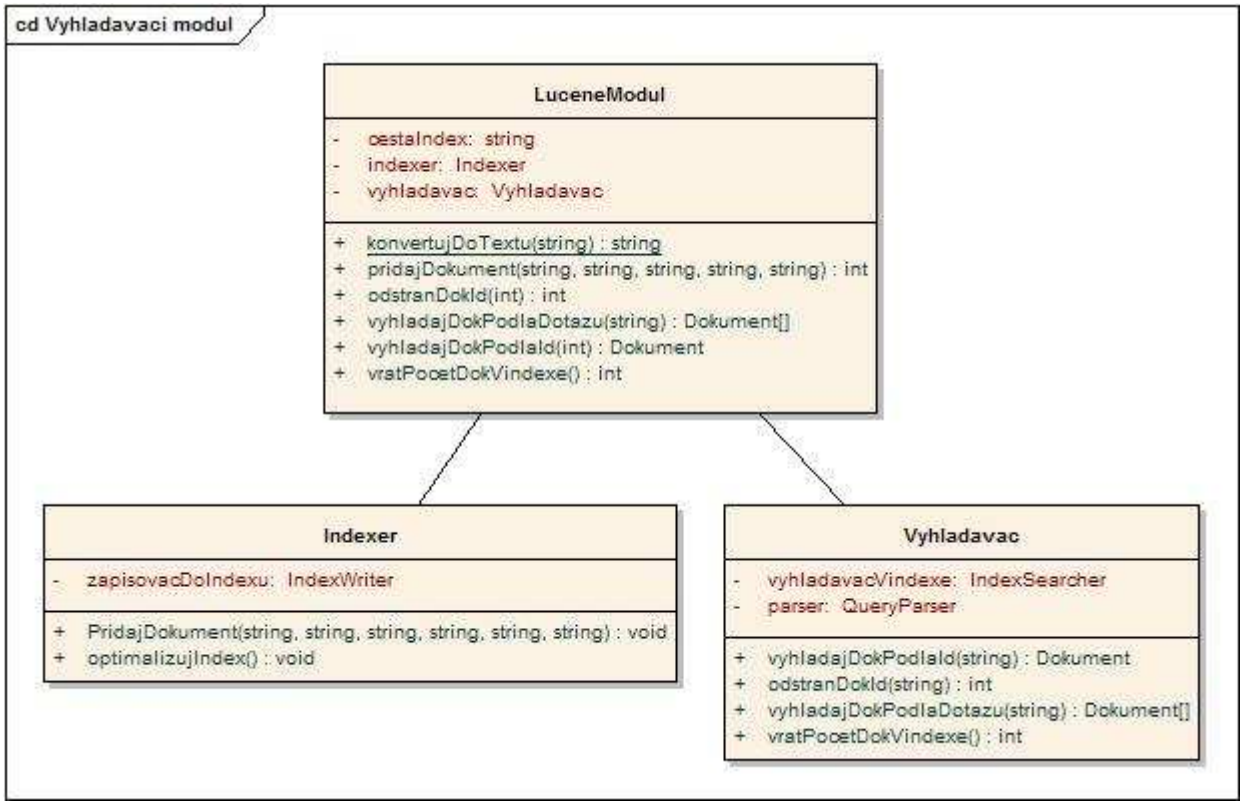
  Funkcia vracia zoznam objektov typu *DokRel*, ktorý obsahuje nasledujúce atribúty:
  - *int dok* – hodnota tohto atribútu reprezentuje identifikačné číslo nájdeného dokumentu
  - *float rel* – obsahuje relevanciu nájdeného dokumentu

  V prípade ak sa nenájde žiadny dokument funkcia vráti prázdny zoznam s nulovou dĺžkou. Funkcia má jeden vstupný parameter, je ním zoznam dokumentov, ktoré boli doteraz nájdené za pomoci iných modulov. K týmto dokumentom sa hľadajú iné dokumenty s ktorými sú prepojené.

4.3 Vyhľadávací modul

Pracuje s vyhľadávacím systémom dotLucene. To znamená, že si vedie index slov používaných v dokumentoch. Tento index je ukladaný v nejakom adresári. Modul je zložený z dvoch častí – *vyhladavac* a *indexer*. Indexer slúži na pridávanie nových súborov do indexu a vyhľadávač na vyhľadávanie v ňom.

Diagram tried modulu je na obr. 30.



Obr. 30. Diagram tried vyhľadávacieho modulu

Trieda LuceneModul

Hlavná trieda modulu. Eviduje cestu k indexu, s ktorým sa pracuje. Pomocou privátnych inštancií tried *Indexer* a *Vyhladavac* s ním pracuje.

Poskytuje tieto metódy:

konvertujDoTextu(string) – transformuje obsah dokumentu do textovej formy pomocou tried projektu Seekafile Server (je umožnená konverzia zo súborov formátov DOC, DOCX, XLS, XLSX, PPT, PDF, RTF, HTM, HTML, TXT)

pridajDokument(string, string, string, string) – zavolá rovnomennú metódu objektu *indexer*, v ktorej pridá dokument do indexu

odstranDokId(int) – zavolá rovnomennú metódu objektu *vyhladavac*, v ktorej odstráni dokument s príslušným identifikačným číslom

vyhladajDokPodlaDotazu(string) – zavolá rovnomennú metódu objektu *vyhladavac*, v ktorej vyhľadá dokumenty vyhovujúce požiadavke zadanej v argumente

vyhladajDokPodlaId(int) – zavolá rovnomennú metódu objektu *vyhladavac*, v ktorej vyhľadá dokument s príslušným identifikátorom

vratPocetDokVindexe() – zavolá rovnomennú metódu objektu *vyhladavac*, v ktorej vráti počet dokumentov zapísaných v indexe

Trieda Indexer

Táto trieda obsahuje objekt triedy *IndexWriter*, ktorý poskytuje funkcie pre zapisovanie do indexu. V metóde *PridajDokument(string, string, string, string, string, string)* zavolá funkciu *AddDocument* triedy *IndexWriter*, ktorá zapíše nový dokument do indexu. Nasleduje ukážka kódu, ktorý pridá dokument do indexu.

```
// vytvorenie dokumentu
public Document VytvorDokument(string id, string nazov, string autor, string kategoria, string
        subor, string text)
{
        Document novyDokument = new Document();

        novyDokument.Add(new Field("id", id, Field.Store.YES, Field.Index.TOKENIZED));
        novyDokument.Add(new Field("nazov", nazov, Field.Store.YES, Field.Index.TOKENIZED));
        novyDokument.Add(new Field("autor", autor, Field.Store.YES, Field.Index.TOKENIZED));
        novyDokument.Add(new Field("kategoria", kategoria, Field.Store.YES,
                Field.Index.TOKENIZED));
        novyDokument.Add(new Field("subor", subor, Field.Store.YES, Field.Index.NO));
        novyDokument.Add(new Field("text", text, Field.Store.NO, Field.Index.TOKENIZED));

        return novyDokument;
}

// pridanie dokumentu do zapisovaca
// 0 ok
// -1 nepodarilo sa prekonvertovat subor do textu
public void PridajDokument(string id, string nazov, string autor, string kategoria, string subor,
        string cestaKsuboru)
{
        otvorZapisovacDoIndexu();

        // prekonvertovanie suboru na text, ktory sa indexuje
        string text = LuceneModul.konvertujDoTextu(cestaKsuboru + subor);

        zapisovacDoIndexu.AddDocument(VytvorDokument(id, nazov, autor, kategoria, subor, text));
        zatvorZapisovacDoIndexu();
}
```

Pre optimalizáciu indexu (spojenie verzií indexov a aktualizáciu) sa použije metóda *optimalizujIndex()*.

Trieda Vyhladavac

Používa sa na narábanie s indexom na úrovni vyhľadávania, s ktorým súvisí aj odstraňovanie dokumentov v indexe. Vytvorí inštanciu triedy *IndexSearcher*, ktorá vyhľadáva v dokumente podľa zadaných požiadaviek. Ďalej obsahuje „parser“ požiadaviek. Používa sa štandardný „parser“ projektu dotLucene.

Vyhľadávanie je zabezpečené prostredníctvom metód:

vyhladajDokPodlaId(string) – vyhľadanie dokumentu s príslušným identifikátorom

vyhladajDokPodlaDotazu(string) – vyhľadanie podľa vyhľadávacieho dotazu podľa syntaxe systému na indexáciu a vyhľadávanie dotLucene

Metódy vracajú objekt, resp. pole objektov štruktúry *Vysledok*. Nižšie sa nachádza kód na vyhľadávanie dokumentov podľa dotazu.

```
// vyhladanie dokumentov podla vyhladavacieho dotazu
public Dokument[] vyhladajDokPodlaDotazu(string dotaz) {
        parser = new QueryParser("text", new StandardAnalyzer());
        try {
                Query query = parser.Parse(dotaz);
                Hits hits = vyhladavacVindexe.Search(query);
                Dokument[] dokumenty = new Dokument[hits.Length()];
                Document tmp;
                for (int i = 0; i < hits.Length(); i++) {
                        tmp = hits.Doc(i);
                        dokumenty[i] = new Dokument(int.Parse(tmp.Get("id").ToString()),
                                tmp.Get("nazov").ToString(),tmp.Get("subor").ToString(),
                                tmp.Get("autor").ToString(),tmp.Get("kategoria").ToString(),
                                hits.Score(i));
                }
                return dokumenty;
        } catch (Lucene.Net.QueryParsers.ParseException) { // nekorektny dotaz
                return null;
        }
}
```

4.4 Modul pre kategórie dokumentov

Kvôli jednoduchšej orientácií v dokumentoch je každému dokumentu priradená kategória, do ktorej patrí. Kategória vypovedá o obsahu dokumentu. Keďže je systém určený pre „heterogénne“ dokumenty rôzneho druhu, formy alebo obsahu, kategórie sú dokumentom priradené explicitne pri pridávaní dokumentu do bázy znalostí. Pre jednoduchosť bol identifikovaný iba malý počet kategórií, ktorý je v prípade nutnosti možné rozšíriť. Ak dokument nespadá do žiadnej z týchto kategórií, priradí sa mu kategória „Nešpecifikovaná“. Zoznam použitých kategórií dokumentov vyzerá nasledovne:

- Nešpecifikovaná
- Analýza
- Architektúra
- Používateľská príručka
- Smernice organizácie
- Technická dokumentácia
- Zdrojový kód Java
- Zdrojový kód C#
- Inštalačná príručka

Kategória, ktorá je dokumentu priradená je uchovávaná v databáze ako atribút v tabuľke dokumentov. Používateľovi je umožnené vyhľadať dokumenty priamo pomocou názvu kategórie v tvare:

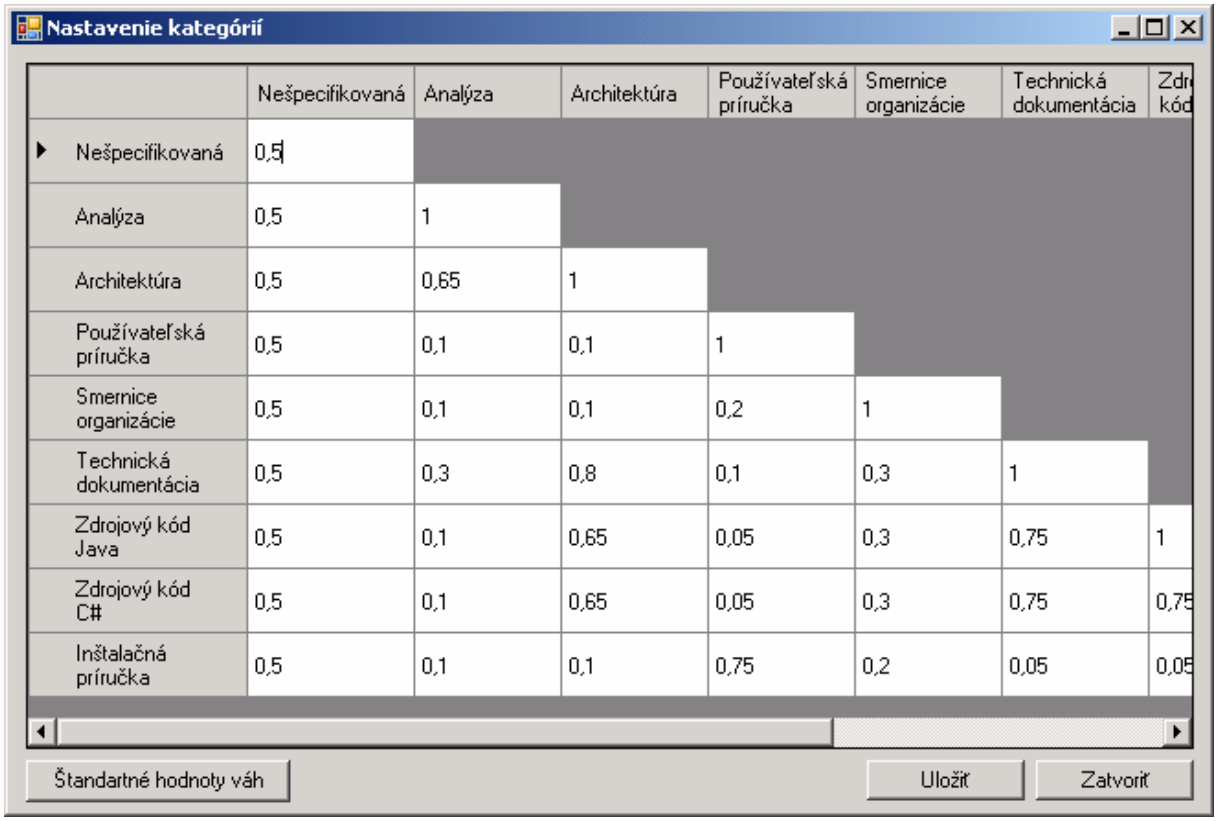
kategoria: [názov kategórie]

Po takto zadanom príkaze vyhľadávací modul *dotLucene* vráti zoznam všetkých dokumentov, ktoré do zadanej kategórie patria.

Druhým možným prípadom využitia modulu kategórií dokumentov je v spojení s vyhľadávaním podľa kľúčových slov v samotnom texte dokumentov. V prípade, že používateľ zadá kľúčové slová a vyberie kategóriu, ktorú chce prednostne nájsť, výsledok bude zoznam dokumentov, v ktorom budú uprednostnené dokumenty zo zvolenej kategórie. Uprednostňovanie sa deje na základe váh medzi samotnými kategóriami. Spojenie každej kategórie s každou inou je reprezentované váhou, teda číslom vyjadrujúcim podobnosť kategórií. Váha je reálne číslo z intervalu $\langle 0,1 \rangle$, kde 0 znamená najmenšiu možnú podobnosť a 1 najväčšiu.

Konkrétne nastavenie váh je viazané na každého používateľa, to znamená, že každý používateľ má možnosť definovať si vlastné podobnosti medzi kategóriami dokumentov. Každý používateľ má potom uložené v databáze svoje informácie o nastavení váh, ktoré sa mu po prihlásení automaticky nastavia. Na začiatku má každý používateľ nastavené tzv. štandardné hodnoty váh. Tieto hodnoty si môže kedykoľvek zmeniť, ale sa aj kedykoľvek vrátiť k pôvodnému nastaveniu.

Nastavenie hodnôt podobnosti dokumentov umožňuje používateľovi jednoduché rozhranie, v ktorom sú váhy podobnosti medzi kategóriami zobrazené v matici. Matica je symetrická, to znamená, že stačí zobraziť hodnoty nad alebo pod hlavnou diagonálou. Váhy štandardných nastavení majú na diagonále hodnotu 1, vyjadrujúcu 100% podobnosť medzi rovnakými kategóriami. „Nešpecifikovaná“ kategória má 50% podobnosť s ktoroukoľvek kategóriou. Na obr. 31 sa nachádza ukážka grafického používateľského rozhrania, ktoré slúži na definovanie podobnosti kategórií používateľom.



Obr. 31. Rozhranie na definovanie podobnosti medzi kategóriami.

Ako už bolo naznačené, v prípade, že sa používateľ rozhodne vyhľadávať dokumenty aj na základe kategórie, pod ktorú spadajú, relevancie zoznamu dokumentov, ktoré vráti vyhľadávací modul *dotLucene* prejdú ešte fázou úpravy. Úprava spočíva v prenásobení danej relevancie váhou spájajúcou kategóriu dokumentu s kategóriou vybranou používateľom:

*celková relevancia = relevancia dokumentu * váha podobnosti kategórií*

Počítanie výsledných váh zabezpečuje trieda *Kategorie*. Na začiatku sa načítajú z databázy hodnoty váh pre konkrétneho používateľa do dvojrozmerného poľa. Po vyhľadaní dokumentov je volaná metóda *vypocetRelevancie()*, ktorá prijíma pole dokumentov a vypočíta pre ne nové váhy. Výber správnej váhy

podobnosti z matice zabezpečuje metóda *vratVahu*, ktorej argumenty sú názvy oboch kategórií, pričom zoznam názvov týchto kategórií je uchovávaný v osobitnom poli.

Modul zohľadňujúci kategórie dokumentov zohráva významnú úlohu v prípade, že používateľ vie presne alebo aspoň približne určiť, do akej kategórie ním hľadaný dokument patrí. Explicitne zadávanie kategórie dokumentu má samozrejme nevýhodu. Automatické zisťovanie kategórie rôznorodých dokumentov však môže byť nesmierne náročné a nepresné.

4.5 Modul modelovania používateľov

Uvedieme, ako sme aplikovali kolaboratívne filtrovanie v našom systéme. Prvou úlohou bolo identifikovať používateľov v systéme. Pri implementácii sme na to navrhli databázovú tabuľku pričom stĺpce prihlasovacie meno a heslo postačujú k úspešnému prihláseniu sa do aplikácie. Prihlasovaním sa jednoducho zistí kto je aktuálnym používateľom (id), to nám postačuje pre generovanie návrhov. Predmetom vyhľadávania budú dokumenty, ktoré sú uložené takisto v databáze. Ďalej sme sa museli rozhodnúť akým spôsobom zaviesť do programu hlasovanie používateľov. Pretože explicitné získavanie dát sa nám javilo ako príliš obťažujúci prístup – vyžaduje to ďalšiu interakciu používateľa a cieľom bolo vytvoriť jednoduchú aplikáciu; použili sme teda implicitné hlasovanie. Zrealizovali sme to tak, že otvorenie dokumentu sa zaznamená do databázy, ktorá obsahuje bunky: identifikátor používateľa, identifikátor dokumentu a počet otvorení. Zmena sa týka počtu otvorení, toto číslo sa zvýši o jedna. Takýto postup sa vykoná vždy pri otvorení výsledku, ktorý sa zobrazí vyhľadaním nejakého dotazu. Hodnotenie dokumentov používateľmi si môžeme predstaviť názornejšie ako maticu, kde riadky sú používatelia, stĺpce sú dokumenty a údaje predstavujú počet otvorení.

Nasledujúcim krokom generovania návrhov je skonštruovanie matice susednosti používateľov (tzv. user-user algoritmus). Prvý semester tímového projektu sme navrhli vzťah ako určiť podobnosť dvoch užívateľov, tento vzorec zmodifikujeme na formulu, ktorá sa nazýva Pearsonova korelácia. V štatistike sa používa tento vzťah na zistenie lineárnej závislosti medzi dvoma náhodnými premennými. Jeho tvar je:

$$D(A,B) = \frac{\sum_{i}^{C(A,B)} \left(V_{A,i_i} - \bar{V}_A \right)^2 \cdot \sum_{i}^{C(A,B)} \left(V_{B,i_i} - \bar{V}_B \right)^2}{\sqrt{\sum_{i}^{C(A,B)} \left(V_{A,i_i} - \bar{V}_A \right)^2 \cdot \sum_{i}^{C(A,B)} \left(V_{B,i_i} - \bar{V}_B \right)^2}} \tag{5}$$

Označenia sú tie isté ako v (4), pribudlo tam naviac hodnotenie $\bar{V}_K$, čo znamená priemerné hodnotenie dokumentov (z celej bázy znalostí) používateľom *K*. Vytvorenie návrhu sme spravili pre niekoľko najpodobnejších používateľov, ich presný počet sa zadáva ako parameter. Rozhodli sme sa používateľovi navrhnúť iba dokumenty, ktoré ešte nevidel, teda ich implicitné hodnotenie aktuálnym používateľom je nulová.
Dokumenty vyhovujúce uvedeným podmienkam zoradíme podľa *vhodnosti* (vhodné navrhnutia), ktoré vypočítame ako suma súčinov hodnotenia dokumentu podobným používateľom a jeho podobnosťou s aktuálne prihláseným.

$$H_{A,D,P} = \sum_{B \in P} D(A,B).V_{B,D} \tag{3}$$

A- je aktuálny používateľ, *P* je množina najpodobnejších používateľov a *D* je dokument, pre ktorý sa počíta hodnotenie, pre *D* platí $V_{A,D} = 0$. Relevancia sa počíta normovaním tejto hodnoty, čo je podelenie hodnoty $H_{A,D,P}$ sumou vhodností vyhovujúcich dokumentov.

Popísaný proces neberie do úvahy výsledky vyhľadania dotazu. To znamená, že vyhľadávaný výraz neovplyvňuje aký dostaneme návrh. Stačí nám jednoducho zobrať si tabuľku hodnotenia dokumentov používateľmi a identifikátor aktuálneho používateľa, z toho už získame návrh tu popísaným spôsobom.

Pôvodne sme do návrhu zahrnuli aj výsledky vyhľadania, znamenalo to vytvorenie matice podobných dokumentov (item-item), kde môžeme použiť napríklad podobnosť kategórií alebo znova Pearsonovu koreláciu. Potom vypočítanie vhodnosti daného dokumentu v sebe zahŕňa aj podobnosť dokumentov aj používateľov.

4.6 Grafické používateľské rozhranie

Nasledujúca časť stručne približuje jednotlivé triedy vytvoreného grafického používateľského rozhrania. Účelom nie je popísať všetky použité existujúce grafické komponenty, ale explicitne vytvorené okná (rozhrania). Nasledujúci model zobrazuje jednotlivé triedy predstavujúce rozhrania a spôsob ich prepojenia (obr. 32).

Obr. 32. Model grafického používateľského rozhrania

Na modely sa nachádzajú dve hlavné triedy – *Administracia* a *Vyhladavanie*. Prvá menovaná trieda slúži ako rozhranie pre administráciu používateľov či dokumentov. Toto rozhranie využíva ďalšiu malú triedu progres. Jedná sa o rozhranie upozorňujúce používateľa na priebeh určitej akcie – môže to byť „upload“ alebo „download“ dokumentu. Toto rozhranie využíva aj druhá spomínaná hlavná trieda s názvom *Vyhladavanie*. Toto rozhranie slúži na používateľovi prevažne na vyhľadávanie dokumentov alebo zobrazovanie výsledkov vyhľadávania. Na prihlásenie sa do systému slúži používateľovi rozhranie definované triedou *Prihlasovanie*. Pomocou neho vyplňuje svoje používateľské meno a heslo. Nastavenie URL servera je možné pomocou rozhrania *Nastavenie servera*. Toto rozhranie je možné spustiť jednak s triedy *Prihlasovanie* ako aj s triedy *Vyhladavanie*. Okrem tohto nastavenia existujú aj ďalšie rozhrania pre nastavenia, konkrétne *Nastavenie vah* na nastavenie váh jednotlivých vyhľadávacích modulov a *Nastavenie kategorii* na nastavenie váh podobnosti kategórií dokumentov. Posledné rozhranie je definované triedou *Napoveda* a predstavuje okno s pomocníkom. Pomocník objasňuje používateľovi spôsob zadávania kľúčových slov.

Rozhrania boli vytvorené pomocou nástroja *Designer* programu *Microsoft Visual C# 2005 Express Edition.* Každé zo spomenutých rozhraní je určené na jednoduchšiu a rýchlejšiu interakciu používateľa so systémom. Rozhrania obsahujú všetky prvky potrebné pre prácu so systémom. Ostávajú však dostatočne jednoduché a intuitívne na používanie.

5 Testovanie

Táto kapitola obsahuje opis testovania, ktoré bolo vykonané po vývoji aplikácie. Na učely externého testovania sa nám podarilo nájsť človeka z firmy HP Slovakia, ktorý pôsobí na oddeleni Consulting and Integration takmer 2 roky. Volá sa Ing. Matej Zavodský a pracuje ako Solution Architect. Matej vypracoval a odovzdal výsledky testovania v požadovanom čase a tie sú uvedené v ďalšej kapitole.

Zoznam testov:

ID	Názov
TS1	Pridanie používateľa
TS2	Odstránenie používateľa
TS3	Prihlásenie používateľa do systému
TS4	Pridanie dokumentu do bázy dát
TS5	Odstránenie dokumentu z bázy dát
TS6	Vyhľadávanie dokumentov
TS7	Otvorenie a stiahnutie nájdeného dokumentu
TS8	Zmena nastavenia váh podobnosti kategórií
TS9	Zmena nastavenia váh modulov

5.1 Testovacie scenáre

ID	TS1	**Názov**	Pridanie používateľa		
Požiadavky		-	**Úroveň splnenia testu**	Musí – ~~Mal by~~ – ~~Mohol by~~	
Rozhranie		Administrácia			
Účel		Pridanie nového používateľa do databázy používateľov			
Vstupné podmienky		-			
Výstupné podmienky		Nový používateľ so zadaným prihlasovacím menom a heslom je pridaný do databázy používateľov			
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**
1	Otvorenie administratívneho okna		Zobrazenie administratívneho okna		Zobrazenie administratívneho okna
2	Vyplnenie prihlasovacieho mena a hesla nového používateľa a kliknutie na tlačidlo „Pridať“		Zobrazenie okna s výsledkom pridávania: Ak používateľ s daným prihlasovacím menom existuje, zobrazenie okna s chybovou správou Ak je pridanie úspešné, informácia o úspechu operácie		Zobrazenie okna s výsledkom pridávania: Ak používateľ s daným prihlasovacím menom existoval, zobrazenie okna s chybovou správou Ak bolo pridanie úspešné, informácia o úspechu operácie

ID	TS2	**Názov**	Odstránenie používateľa		
Požiadavky		-	**Úroveň splnenia testu**	Musí – ~~Mal by~~ – ~~Mohol by~~	
Rozhranie		Administrácia			
Účel		Odstránenie používateľa z bázy dát			
Vstupné podmienky		Existencia používateľa v systéme			
Výstupné podmienky		Používateľ odstránený z bázy dát			
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**
1	Spustenie administrátorského rozhrania		Zobrazenie administrátorského rozhrania		Zobrazenie administrátorského rozhrania
2	Vyplnenie textového poľa „Login" a stlačenie tlačidla „Odstrániť" v panely „Používatelia"		Správa o úspešnom odstránení používateľa z bázy dát		Správa o úspešnom odstránení používateľa z bázy dát

ID	TS3	**Názov**	Prihlásenie používateľa do systému		
Požiadavky		-	**Úroveň splnenia testu**	Musí – ~~Mal by~~ – ~~Mohol by~~	
Rozhranie		Prihlasovanie			
Účel		Prihlásenie sa do systému			
Vstupné podmienky		-			
Výstupné podmienky		Prihlásenie používateľa do systému (v prípade správneho prihlasovacieho mena a hesla)			
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**
1	Spustenie rozhrania na prihlasovanie sa používateľa do systému		Zobrazenie rozhrania na prihlasovanie sa používateľa do systému		Zobrazenie rozhrania na prihlasovanie sa používateľa do systému
2a	Vyplnenie správneho (existujúceho) používateľského mena + hesla a stlačenie tlačidla „OK"		Úspešné prihlásenie – spustenie rozhrania pre vyhľadávanie dokumentov		Úspešné prihlásenie – spustenie rozhrania pre vyhľadávanie dokumentov
2b	Vyplnenie nesprávneho (neexistujúceho) používateľského mena + hesla a stlačenie tlačidla „OK"		Chybové hlásenie o nesprávnosti používateľského mena alebo hesla		Chybové hlásenie o nesprávnosti používateľského mena alebo hesla

ID	TS4	**Názov**	Pridanie dokumentu do bázy dát		
Požiadavky		-	**Úroveň splnenia testu**	Musí – ~~Mal by~~ – ~~Mohol by~~	
Rozhranie		Administrácia			
Účel		Pridanie dokumentu do bázy dát			
Vstupné podmienky		Existujúci dokument s právami na čítanie			
Výstupné podmienky		Dokument pridaný do bázy dát			

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Spustenie administrátorského rozhrania	Zobrazenie administrátorského rozhrania	Zobrazenie administrátorského rozhrania
2	Kliknutie na tlačidlo „Vyhľadať" v časti pridanie dokumentu	Zobrazenie dialógu pre výber súboru	Zobrazenie dialógu pre výber súboru
3	Výber dokumentu na pridanie a potvrdenie jeho výberu	Zatvorí sa dialógové okno pre výber súboru a v poli „Súbor" sa zobrazí cesta k vybranému dokumentu	Zatvorenie dialógového okna pre výber súboru a zobrazenie cesty k vybranému dokumentu v poli „Súbor"
4	Vyplnenie údajov o dokumente a potvrdenie jeho pridania kliknutím na tlačidlo „Pridať"	Zobrazenie vyskakovacieho okna „Posielanie súboru" počas celej doby pridávania dokumentu do bázy dát, po dokončení zobrazenie okna o úspešnom pridaní	Zobrazenie vyskakovacieho okna „Posielanie súboru" počas celej doby pridávania dokumentu do bázy dát, po dokončení zobrazenie okna o úspešnom pridaní

ID	TS5	**Názov**	Odstránenie dokumentu z bázy dát		
Požiadavky		-	**Úroveň splnenia testu**	Musí – ~~Mal by~~ – ~~Mohol by~~	
Rozhranie		Administrácia			
Účel		Odstránenie dokumentu z bázy dát			
Vstupné podmienky		Dokument existujúci v báze dát			
Výstupné podmienky		Dokument odstránený z databázy			

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Otvorenie administratívneho okna	Zobrazenie administratívneho okna	Zobrazenie administratívneho okna
2	Vyplnenie textového poľa „Kľúčové slová" dotazom, ktorý nájde dokument existujúci v báze dát a kliknutie na tlačidlo „Vyhľadať"	Naplnenie tabuľky dokumentmi, ktoré spĺňajú kritériá vyhľadávania	Naplnenie tabuľky dokumentmi, ktoré spĺňajú kritériá vyhľadávania
3	Označenie jedného alebo viacerých dokumentov (pomocou držania klávesy CTRL)	Zvýraznenie označených dokumentov	Zvýraznenie označených dokumentov
4	Kliknutie na tlačidlo „Odstrániť"	Zvýraznené dokumenty zmiznú z tabuľky výsledkov vyhľadávania	Zvýraznené dokumenty zmiznú z tabuľky výsledkov vyhľadávania

ID	TS6	**Názov**	Vyhľadávanie dokumentov		
Požiadavky		TS3	**Úroveň splnenia testu**	Musí – ~~Mal by~~ – ~~Mohol by~~	
Rozhranie		Vyhľadávanie v báze znalostí			
Účel		Overenie, či systém vyhľadá dokumenty podľa požiadaviek používateľa			
Vstupné podmienky		Úspešné prihlásenie do používateľskej časti systému (rozhranie Vyhľadávanie), existujúce dokumenty v báze dát			
Výstupné podmienky		-			

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Výber modulov, ktoré sa majú použiť vo vyhľadávaní	Označenie vybraných modulov pre vyhľadávanie	Označenie vybraných modulov pre vyhľadávanie
2	Vyplnenie textového poľa „Kľúčové slová" dotazom pre vyhľadávací modul podľa zodpovedajúcej syntaxe a kliknutie na tlačidlo „Vyhľadať"	Naplnenie tabuľky výsledkov dokumentmi, ktoré spĺňajú požiadavky vyhľadávania a vypísanie počtu nájdených dokumentov zhodné s počtom dokumentov v tabuľke výsledkov	Naplnenie tabuľky výsledkov dokumentmi, ktoré spĺňajú požiadavky vyhľadávania a vypísanie počtu nájdených dokumentov zhodné s počtom dokumentov v tabuľke výsledkov

ID	TS7	**Názov**	Otvorenie a stiahnutie nájdeného dokumentu		
Požiadavky		TS3	**Úroveň splnenia testu**	Musí – ~~Mal by~~ – ~~Mohol by~~	
Rozhranie		Vyhľadávanie v báze znalostí			
Účel		Overenie otvorenia a stiahnutia nájdeného dokumentu			
Vstupné podmienky		Úspešné prihlásenie do používateľskej časti systému (rozhranie Vyhľadávanie), nájdené dokumenty v tabuľke výsledkov			
Výstupné podmienky		Otvorený alebo stiahnutý označený dokument			

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Označenie dokumentu, ktorý sa má stiahnuť alebo otvoriť kliknutím myši na riadok tabuľky, v ktorom sa dokument nachádza	Zvýraznenie označeného dokumentu	Zvýraznenie označeného dokumentu
2a	Kliknutie na tlačidlo „Otvoriť dokument"	Zobrazenie vyskakovacieho okna „Sťahovanie dokumentu", otvorenie zvoleného dokumentu v programe priradenom operačným systémom	Zobrazenie vyskakovacieho okna „Sťahovanie dokumentu", otvorenie zvoleného dokumentu v programe priradenom operačným systémom
2b	Kliknutie na tlačidlo „Uložiť dokument"	Zobrazenie vyskakovacieho okna „Sťahovanie dokumentu", otvorenie dialógového okna na uloženie dokumentu	Zobrazenie vyskakovacieho okna „Sťahovanie dokumentu", otvorenie dialógového okna na uloženie dokumentu
3b	Zvolenie, kde sa má dokument uložiť a kliknutie na tlačidlo „Uložiť"	Zatvorenie okna na uloženie súboru a dokument je uložený na zvolenom mieste	Zatvorenie okna na uloženie súboru a dokument je uložený na zvolenom mieste

ID	TS8	**Názov**	Zmena nastavenia váh podobnosti kategórií		
Požiadavky		TS3	**Úroveň splnenia testu**	Musí – ~~Mal by~~ – ~~Mohol by~~	
Rozhranie		Nastavenie kategórií			
Účel		Zadefinovanie vlastných váh podobnosti kategórií dokumentov			
Vstupné podmienky		Úspešne prihlásený používateľ do systému			
Výstupné podmienky		Zmena nastavenia váh podobnosti kategórií			

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Otvorenie rozhrania výberom v ponuke Nastavenia -> Kategórie	Zobrazenie rozhrania na „Nastavenie kategórií"	Zobrazenie rozhrania na „Nastavenie kategórií"
2a	Zmena niektorej váhy na inú hodnotu mimo intervalu <0,1> a stlačenie tlačidla „Uložiť"	Zobrazenie chybovej správy o zadaní nekorektnej hodnoty. Pôvodná hodnota sa nezmení.	Zobrazenie chybovej správy o zadaní nekorektnej hodnoty. Pôvodná hodnota sa nezmení.
2b	Zmena niektorej váhy na inú hodnotu z intervalu <0,1> a stlačenie tlačidla „Uložiť	Zobrazenie hlásenia o úspešnom uložení váh.	Zobrazenie hlásenia o úspešnom uložení váh.
3	Kliknutie na tlačidlo „Zatvoriť"	Zatvorenie rozhrania na „Nastavenie kategórií"	Zatvorenie rozhrania na „Nastavenie kategórií"
4	Otvorenie rozhrania výberom v ponuke Nastavenia -> Kategórie a skontrolovanie zmenenej hodnoty	Zmenená hodnota na danom mieste.	Zmenená hodnota na danom mieste.

ID	TS9	**Názov**	Zmena nastavenia váh modulov		
Požiadavky		TS3	**Úroveň splnenia testu**	Musí – ~~Mal by~~ – ~~Mohol by~~	
Rozhranie		Nastavenie váh			
Účel		Zadefinovanie vlastných váh modulov			
Vstupné podmienky		Úspešne prihlásený používateľ do systému			
Výstupné podmienky		Zmena nastavenia váh modulov			

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Otvorenie rozhrania výberom v ponuke Nastavenia -> Váhy	Zobrazenie rozhrania na „Nastavenie váh"	Zobrazenie rozhrania na „Nastavenie váh"
2	Zmena niektorej váhy na inú hodnotu mimo intervalu a stlačenie tlačidla „Uložiť"	Normalizovanie váh a zobrazenie hlásenia o úspešnom uložení váh	Normalizovanie váh a zobrazenie hlásenia o úspešnom uložení váh
3	Kliknutie na tlačidlo „Zatvoriť"	Zatvorenie rozhrania na „Nastavenie váh"	Zatvorenie rozhrania na „Nastavenie váh"
4	Otvorenie rozhrania výberom v ponuke Nastavenia -> Váhy a skontrolovanie zmenenej hodnoty	Zobrazenie zmenených váh	Zobrazenie zmenených váh

6 Zhodnotenie

Dokumentácia je výsledkom opisu práce tímu na projekte. Zachytáva jednotlivé fázy. projektu a ponúka celkový pohľad na vytvorený systém. Okrem častí, ktoré sú uvedené v dokumentácií boli počas projektu vyprodukované aj iné samostatné dokumenty pre interné účely, ktoré napomohli pri riešení projektu.

Pozitívnou motiváciou tímu bol fakt, že projekt bude mať aj praktické využitie, keď poslúži ďalším študentom pri práci na projektoch podobného charakteru.

Tímový projekt hodnotíme ako pozitívnu skúsenosť, ktorá nás všetkých posunula ďalej. Preto by sme sa radi na záver poďakovali všetkým, ktorí nám pomohli zvládnuť túto náročnú skúšku vytrvalosti a tímovej spolupráce.

7 Použitá literatúra

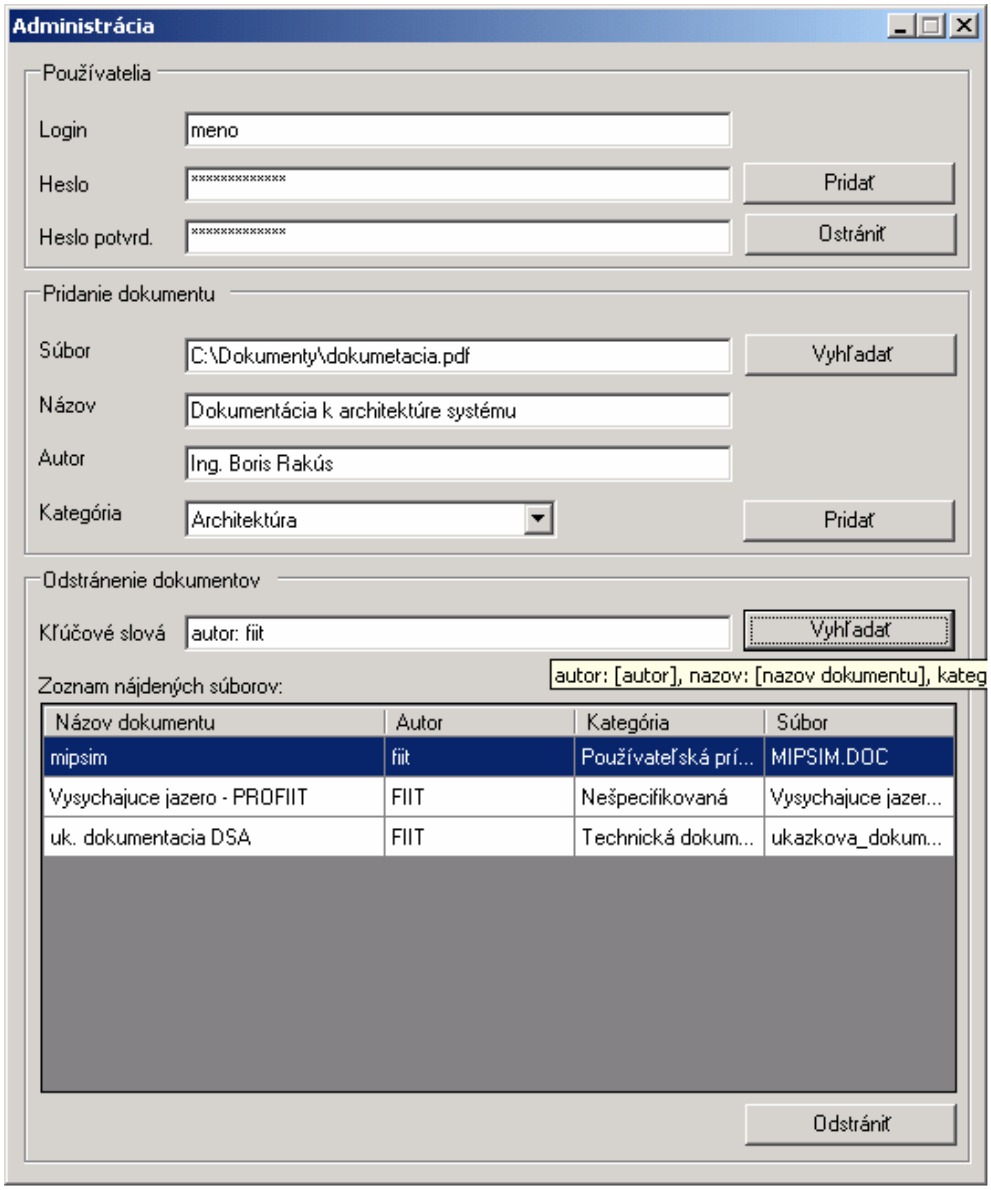
1. Znalostný manažment na báze technológie .NET (Tímový projekt) – dokumentácia k projektu Lucky Number 7, http://www2.dcs.elf.stuba.sk/TeamProject/2006/team07/public_html/ (14.11.2007)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining associations rules. In: J.B. Bocca, M.Jarke. C.Zaniolo (eds.), Proc. of 20th VLDB Conference, Santiago de Chile, 12-15 September 1994, Morgan Kaufmann, pp. 487-499.
3. Bieliková, M., Návrat, P., a kol.: Štúdie vybraných tém softvérového inžinierstva 2. Slovenská technická univerzita, 2006.
4. Duffner, S.: A Knowledge Portal for Multi-Project Management. Karlsbad, 2000. 90 s. Diplomová práca na Univerzite Karlshure. Vedúci prace: prof. Dr. Studer.
5. Fröschl, Christoph: User Modeling and User Profiling in Adaptive E-learning Systems, diplomová práca na Technickej Univerzite Graz, November 2005
6. Gudivada, V. N. et al.: Information Retrieval on the World Wide Web, IEEE Internet Computing, September-October 1997, pp. 58-68.
7. Konchady, M.: Text Mining Application Programming. Charles River Media, Boston, 2006.
8. Paralič, J.: Manažment znalostí – podklady k prednáškam, http://www.tuke.sk/paralicj/mz.html (15.11.2006)
9. Razmerita, L., Angehrn, A., Maedche, A.: Ontology-based User Modeling for Knowledge Management Systems, INSEAD, CALT-Centre of Advanced Learning Technologies
10. Seekafile - flexible indexing server, http://sourceforge.net/project/showfiles.php?group_id=143466/ 10.11.2007)
11. http://lucene.apache.org/java/docs/features.html (8.11.2007)
12. http://www.codeproject.com/aspnet/DotLuceneSearch.asp (10.11.2007)
13. Gudivada, V. N. et al.: Information Retrieval on the World Wide Web, IEEE Internet Computing, September-October 1997, pp. 58-68.
14. M. Rashid, G. Karypis, J. Riedl: Influence in Ratings-Based Recommender Systems: An Algorithm-Independent Approach, Proceedings of SIAM 2005 Data Mining Conference, 2005
15. Tuzhilin, A. and Adomavicius; Integrating user behaviour and collaborative methods in recommender systems, CHI Workshop, 1999

Príloha A – Používateľská príručka

Pre používateľsky prijateľnejšie narábanie je v systéme implementované grafické používateľské rozhranie. Na komunikáciu so systémom boli vyhradené dve hlavné rozhrania. Prvé sa týka administrácie a zahŕňa činnosti spojené s pridávaním nových používateľov do databázy, pridávaním nových dokumentov do databázy či ich odstraňovaním. Druhé hlavné rozhranie sa týka samotného vyhľadávania dokumentov.

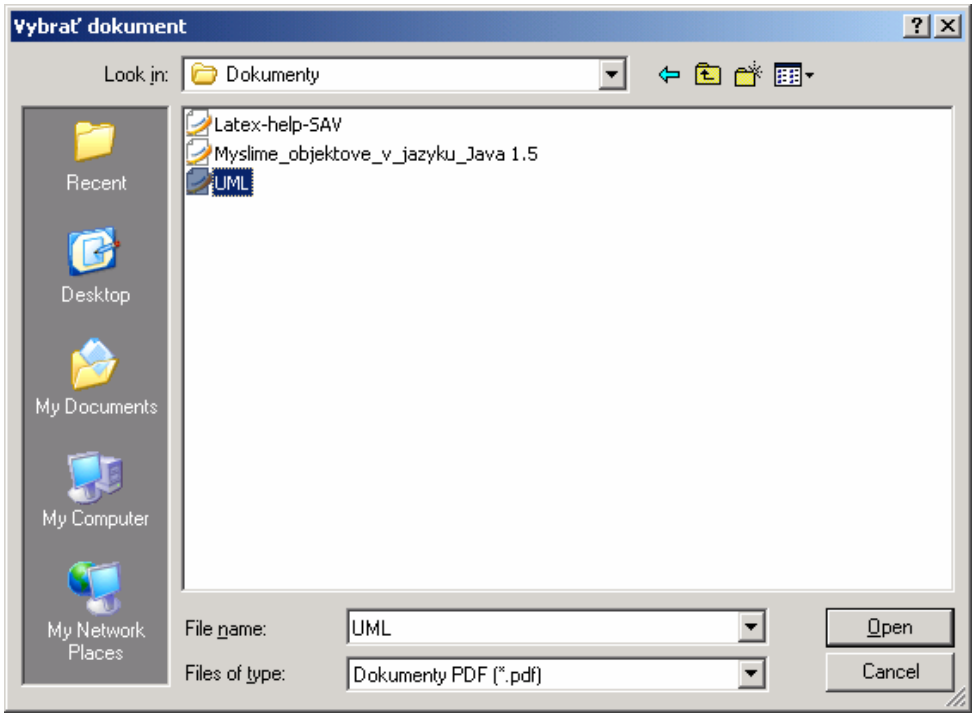
A.1 Rozhranie na administráciu

Rozhranie na administráciu je rozdelené do troch častí (obr. 1). Prvá časť sa týka administrácie používateľov systému. Táto časť je určená na pridávanie nových, alebo odstraňovanie už existujúcich používateľov. Na pridanie používateľa do databázy slúži metóda *pridajPouzivatela(login, passwd)*. Na jeho odstránenie je volaná metóda *odstranPouzivatela(login)*.



Obr. 1. Rozhranie na administráciu

Druhý panel *Pridanie dokumentu* slúži na pridávanie nových dokumentov do bázy znalostí. Na vyhľadávanie a vybratie lokálneho súboru slúži nástroj *FileDialog* zobrazený na obr. 2. Pri pridávaní súboru je ďalej možné vyplniť kolonky *Názov*, *Autor* a tiež priradiť dokumentu kategóriu. Pridanie dokumentu je zabezpečené volaním metódy *pridajDokument()*, ktorej argumentom je okrem spomenutých atribútov aj samotný súbor v podobe poľa bajtov (*byte[]*). Dáta, ktoré sú v tomto poli posielané sú ukladané do databázy. Ostatné atribúty tvoria metadáta o súbore.

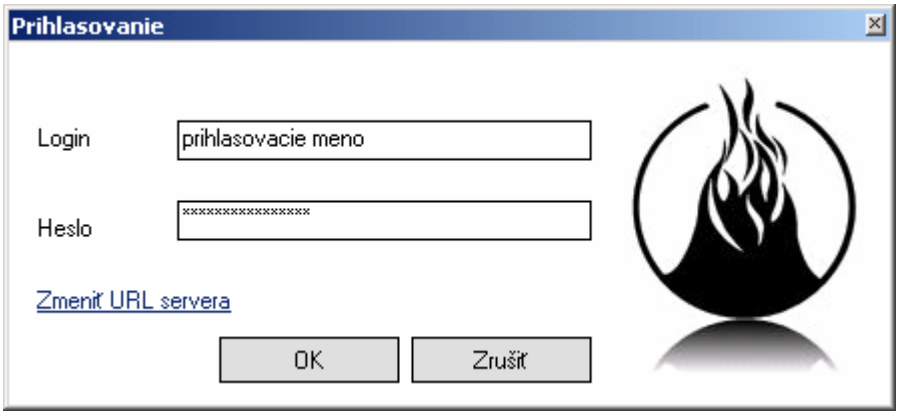


Obr. 2. Ukážka nástroja *FileDialog*

Posledný panel slúži na odstraňovanie dokumentov z bázy znalostí. V prípade, že potrebuje administrátor odstrániť určitý dokument (prípadne naraz viac dokumentov), musí najskôr potrebný dokument v báze znalosti nájsť. Dokumenty môže vyhľadať pomocou kľúčových slov zadaných do textového panelu. Takto zadané slová sa budú hľadať v samotnom texte dokumentu. V prípade, že chce administrátor hľadať kľúčové slová v atribútoch dokumentu, musí ich zadať v tvare „názov atribútu: kľúčové slová“. Podporovanými atribútmi sú *„autor“*, *„nazov“*, *„kategoria“* a *„subor“*. Po vyhľadaní dokumentov stačí potrebné dokumenty označiť a stlačiť tlačidlo „Odstrániť“.

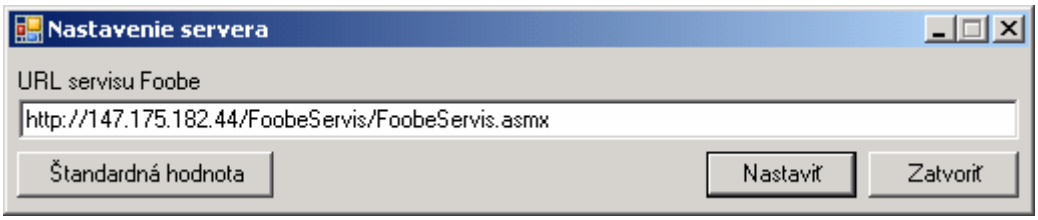
A.2 Rozhranie na prihlasovanie

Predtým ako používateľ začne vyhľadávať dokumenty v báze znalostí, musí sa do systému prihlásiť pod svojím meno, ktoré bolo už predtým zaregistrované v systéme. Prihlasovacie meno slúži ako primárny kľúč, to znamená, že je jedinečné pre každého používateľa. Na prihlasovanie sa do systému slúži používateľské rozhranie zobrazené na obr. 3. Do textových polí užívateľ zadáva svoje prihlasovacie meno (Login) a heslo. Po správnom vyplnení textových polí stačí používateľovi stlačiť tlačidlo „OK“. Tlačidlo „Zrušiť“ slúži na ukončenie aplikácie.



Obr. 3. Rozhranie na prihlasovanie sa do systému

V prípade nesprávneho vyplnenia údajov a pokusu o prihlásenie, systém používateľa upozorní chybovou správou „Zlý login alebo password!“. Okrem toho môže tiež dôjsť k chybe pri komunikácií so serverom. V takom prípade môže byť nesprávne nastavená URL adresa servera. Používateľ môže túto adresu zmeniť po kliknutí na odkaz „Zmeniť URL servera“. Na nastavenie adresy slúži rozhranie zobrazené na obr. 4. Okrem prepísania adresy je k dispozícií aj prednastavená adresa.



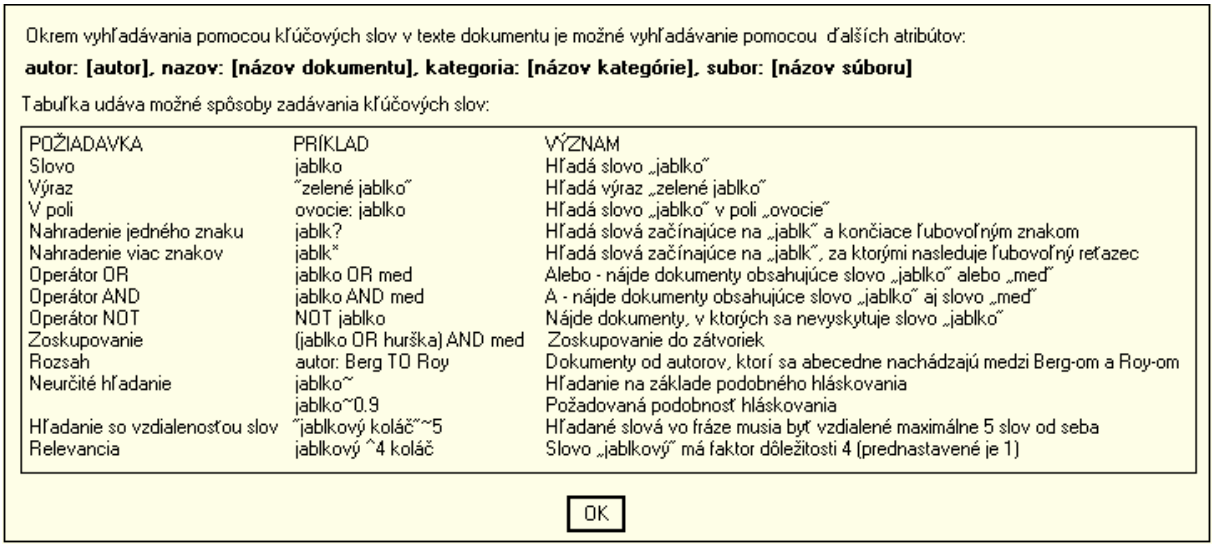
Obr. 4. Rozhranie na nastavenie adresy servera

A.3 Rozhranie na vyhľadávanie dokumentov

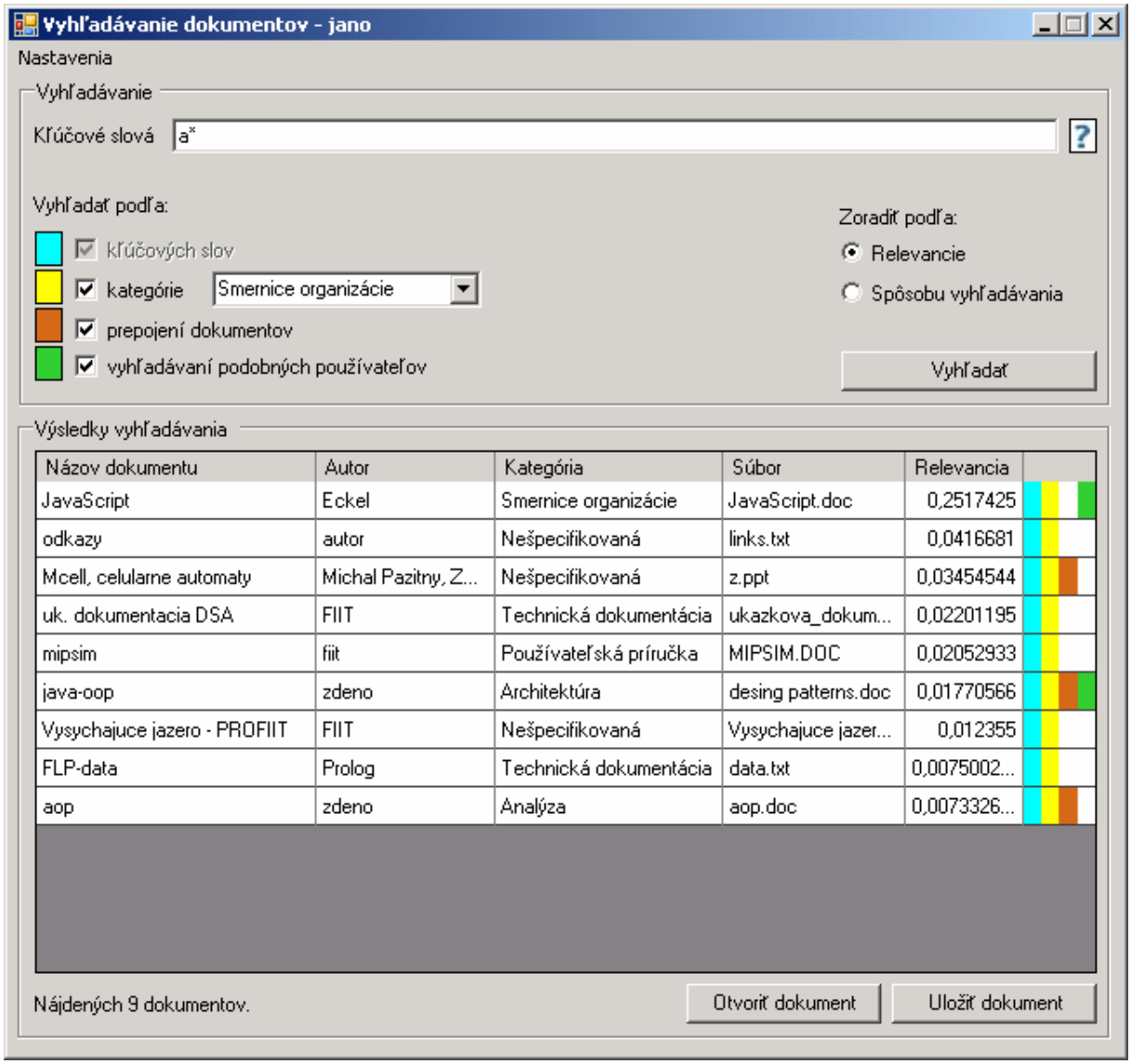
Po úspešnom prihlásení sa používateľovi ukáže hlavné používateľské rozhranie určené na vyhľadávanie dokumentov v báze znalostí. Ukážka tohto rozhrania sa nachádza na obr. 5. Prihlasovacie meno používateľa je zobrazené v názve okna. Okno je rozdelené do dvoch hlavných častí. Prvá časť slúži na zadávanie parametrov a nastavení. Druhá časť slúži prevažne na zobrazovanie výsledkov vyhľadávania.

Základom pri vyhľadávaní dokumentov sú kľúčové slová, ktoré sa zadávajú do textového poľa. Pri zadávaní kľúčových slov má používateľ viacero možností. Po kliknutí na otáznik napravo od textového poľa sa zobrazí okno s pomocníkom (obr. 6). V tejto nápovede je sú vysvetlené možnosti zadávania kľúčových slov vysvetlené.

Pri vyhľadávaní je možné aktivovať jednotlivé moduly ich označením (zaškrtnutím). Moduly sú označené farebne, čo uľahčuje identifikáciu spôsobu vyhľadania. V prípade, že chce používateľ vyhľadávať aj na základe kategórie dokumentu, okrem označenia modulu si musí vybrať aj z ponuky žiadanú kategóriu.



Obr. 5. Pomocník – možnosti zadávania kľúčových slov



Obr. 6. Ukážka používateľského rozhrania na vyhľadávanie dokumentov

Po výbere vyhľadávacích modulov a prípadnom zadaní kľúčových slov sa proces vyhľadávania spustí stlačením klávesy *Enter* alebo kliknutím na tlačidlo „Vyhľadať“. Proces hľadania spustí metóda *vyhladajDokumenty()* s argumentmi *klucoveSlova*, *kategoria*, *login* a *sposobVyhladania* (definuje aktivované moduly).

Ďalší panel okna má názov „Výsledky vyhľadávania“. Jeho hlavným prvkom je tabuľka, v ktorej sa po spustení vyhľadávania zobrazia nájdené dokumenty. Tabuľka má šesť stĺpcov pre identifikovanie šiestich atribútov (názov dokumentu, autor, kategória, súbor, relevancia a označenie modulov, ktoré sa podielali na nájdení výsledku). V ľavej spodnej časti obrazovky sa zobrazuje počet nájdených dokumentov.

Vyhľadané dokumenty zobrazené v tabuľke je možné otovriť na prezeranie alebo uložiť na disk. V prípade uloženia dokumentu (tlačidlo „Uložiť“) je dokument poslaný zo servera na klientský počítač. Stiahnutie dokumentu je volané metódou *stiahniDokument()*, ktorej argumentmi je používateľské meno a identifikačné číslo dokumentu. Pri posielaní súboru klinetskej aplikácií je znemožnené používanie lavného okna. Na sťahovanie súboru používateľa upozorní samostatné okno zobrazené na obr. 7. Po stiahnutí dokumentu sa zobrazí *FileDialog*, pomocou ktorého môže používateľ zvoliť cestu alebo zmeniť názov súboru.

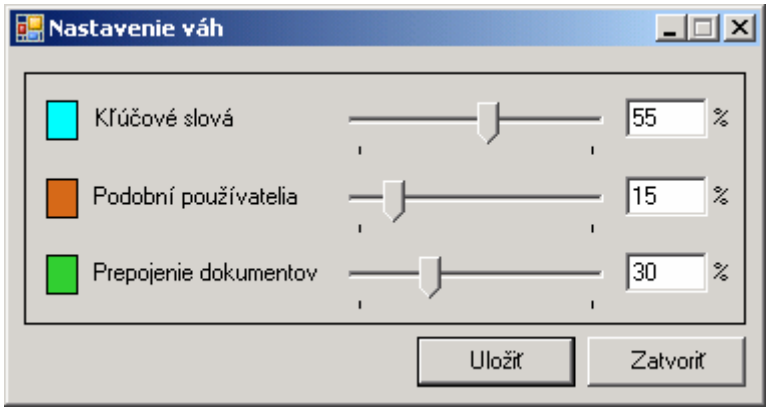
Obr. 7. Okno signalizujúce sťahovanie súboru

Podobným spôsobom je realizovaná aj možnosť priameho otvorenia dokumentu. Dokument je rovnako potrebné najskôr stiahnuť, v tomto prípade však do adresára dočasných súborov. Cesta k tomuto adresáru je získaná pomocou metódy *System.IO.Path.GetTempPath()*. Po stiahnutí a uložení dokumentu do tohto adresára je primo spustená príslušná aplikácia umožňujúca prehliadanie dokumentu.

A.4 Rozhranie na nastavenie váh modulov

Ako už bolo spomenuté, vyhľadávanie je zabezpečené pomocou používateľom vybraných modulov. Modulom je možné nastaviť váhu. Nastavenie váh modulov je uložené pre každého používateľa zvlášť. Po nastavení váh prebieha normovanie tak, aby tvoril súčet nastavených hodnôt 100%. Na začiatku má každý používateľ nastavené štandartné váhy – každý z troch modulov má váhu 33,33..%.

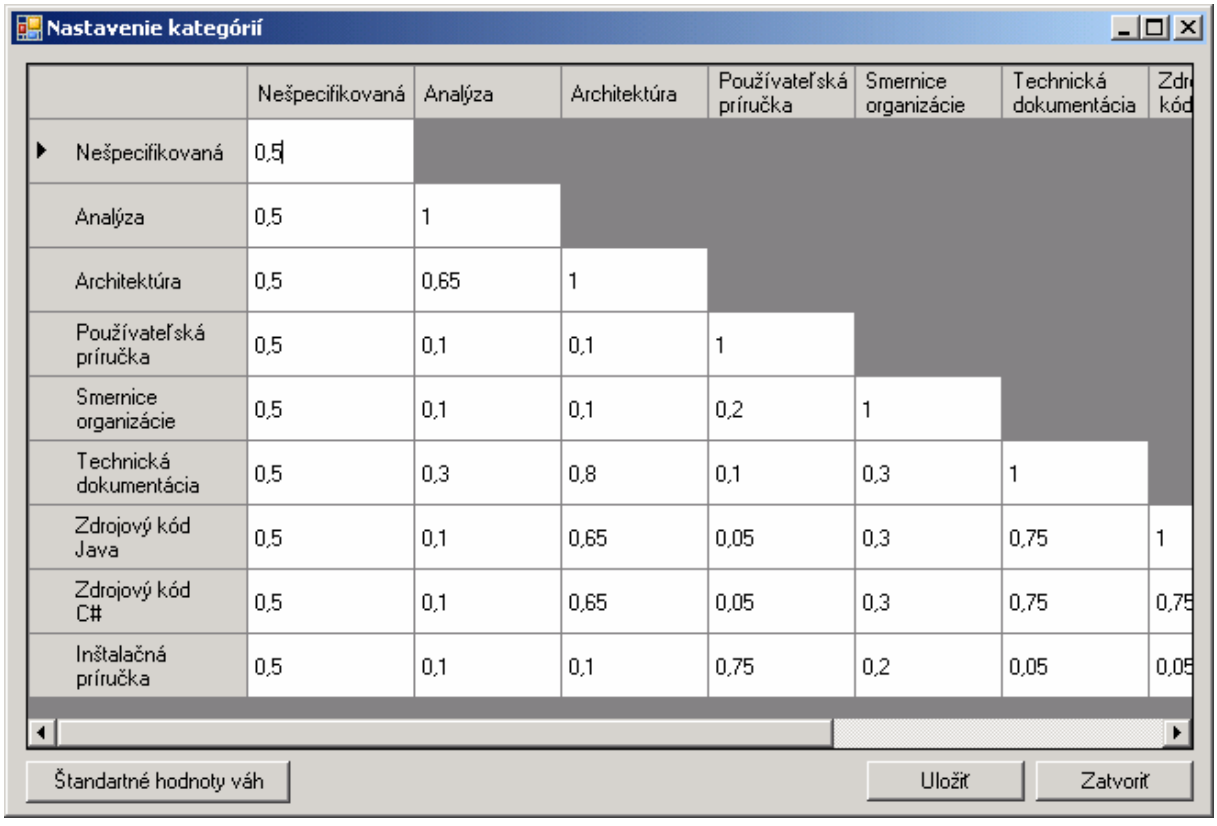
Nastavenie váh slúži na upravovanie celkovej relevancie nájdených dokumentov podľa používateľovej potreby. Zoznam dokumentov s výslednou relevanciou teda ešte prechádza zmenou podľa nastavených váh. Ak je aktivovaný daný vyhľadávací modul a nájde niektorý z dokumentov, celková relevancia tohto dokumetu bude násobená váhou, ktorá prislúcha modulu. Na obr. 8 je zobrazené rozhranie, ktoré umožňuje nastavovanie váh modulov. Po stlačení tlačidla „Uložiť“ nastane uloženie váh do databázy pomocou metódy *zmenVahyModulov()*. Argumentami metódy sú jednotlivé váhy spolu s používateľským menom.



Obr. 8. Rozhranie na nastavenie váh modulov

A.5 Rozhranie na nastavenie váh podobnosti kategórií

Nasledujúce rozhranie sa viaže k jednému s modulov, konkrétne k modulu vyhľadávania na základe kategórie dokumentu (obr. 9). Rozhranie umožňuje používateľovi meniť váhy podobnosti medzi kategóriami, pričom opäť platí, že každý používateľ môže mať vlastné váhy. Zmeny používateľom nastavených váh je možné opätovane vrátiť po stlačení tlačidla „Štandardné hodnoty váh“.



Obr. 9. Rozhranie na nastavenie váh podobnosti kategórií

Váhy sú zobrazené ako reálne čísla v tabuľke. Nakoľko je matica s váhami dokumentov symetrická, tabuľka obsahuje iba bunky pod hlavnou diagonálou. O zaznamenanie váh sa stará metóda *zmenVahyKategorii()*, ktorej je odovzdané používateľské meno spolu so zoznamom vvšetkých váh.

Príloha B – Inštalačná príručka

Táto príručka obsahuje postup, ako nainštalovať systém pre znalostný manažment *Foobe*.

Produkt *Foobe* sa skladá z dvoch častí: servera a klientských aplikácií. Server poskytuje webovú službu, s ktorou komunikujú klienti. Webová služba má názov *FoobeServis* a spolupracuje so systémom pre spravovanie bázy znalostí, databázou a modulmi rozširujúcimi funkcionality systému znalostného manažmentu. Klientov, ktorí poskytnú používateľom grafické používateľské rozhranie na administráciu bázy znalostí a vyhľadávanie dokumentov v nej, budeme označovať *FoobeGui*. V príručke sú vysvetlené inštalácie oboch častí v jednotlivých kapitolách.

B.1 Inštalácia webovej služby a systému znalostného manažmentu

Webová služba *FoobeServis* bude spúšťaná na serveri systému, najlepšie na nepretržite prístupnom počítači v rámci siete, v ktorej sa bude systém používať. Je určená pre server IIS. Spolupracuje s databázou MS SQL Server 2005. Poskytuje funkcie pre spravovanie systému znalostného manažmentu *Foobe*.

Pre inštaláciu je potrebné splniť nasledujúce požiadavky:
- nainštalovaný IIS server
- nainštalovaný MS SQL Server 2005
- práva meniť nastavenia IIS servera aj MS SQL Servera 2005

Kroky inštalácie:

1. Rozbaľte obsah súboru *FoobeServer.zip* do ľubovoľného adresára na serveri IIS.

2. Adresáru *App_Data* nastavte práva na zápis aj čítanie.

3. Adresáru *App_Data* zmeňte nastavenia v rámci servera IIS nasledovne:

   3.1. Otvorte nástroj na správu servera IIS
   Ovládací panel (Control Panel) → *Nástroje na správu (Administrative Tools)* → *Internet Information Services*

   3.2. Otvorte nastavenia tohto adresára, ako na obr. 1.

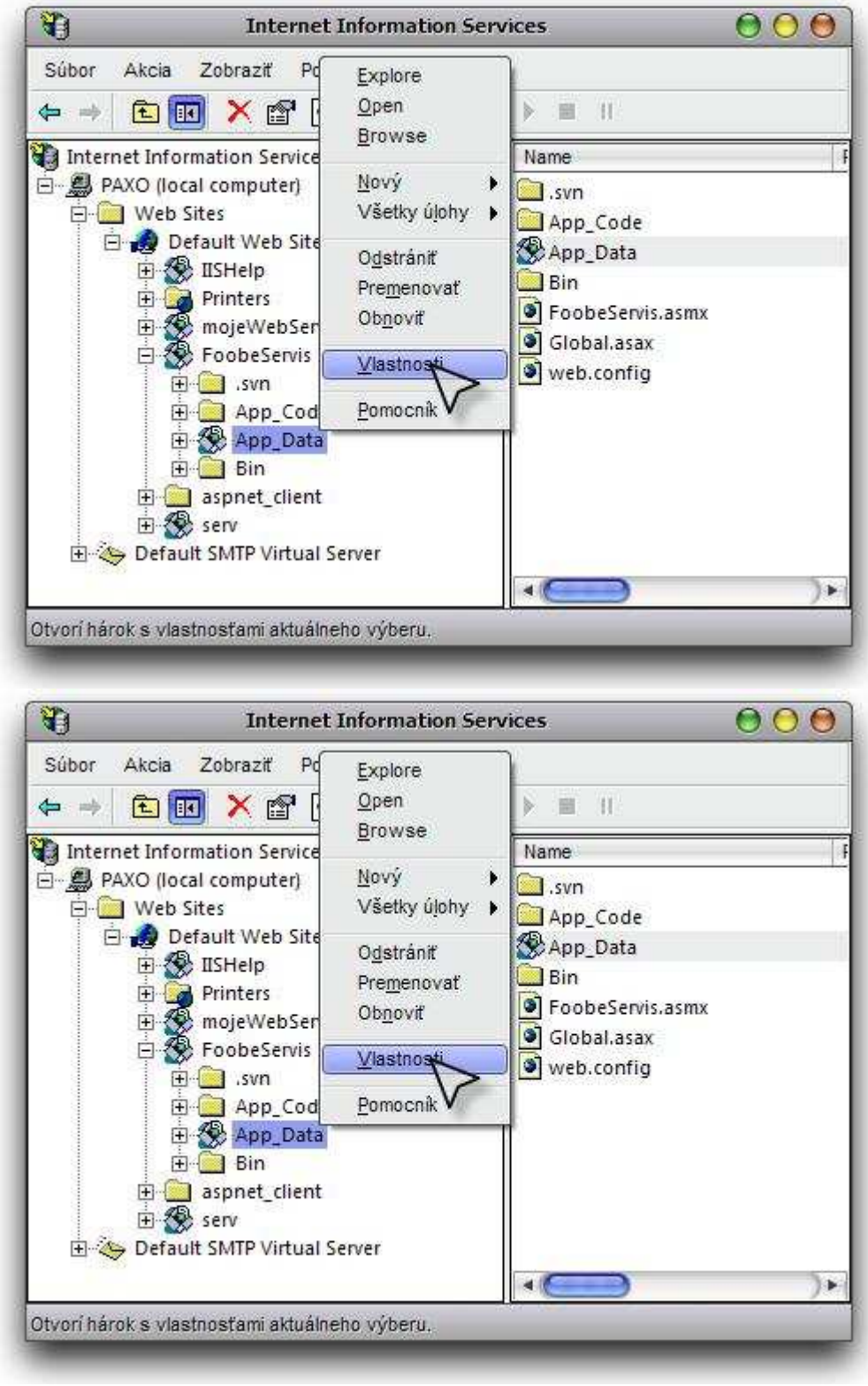
   3.3. Nastavte vlastnosti podľa obr. 2.

   3.4. Nastavte používateľa podľa obr. 3 – zmeňte meno používateľa *'PAXO'* na používateľské meno, pod ktorým beží IIS server.

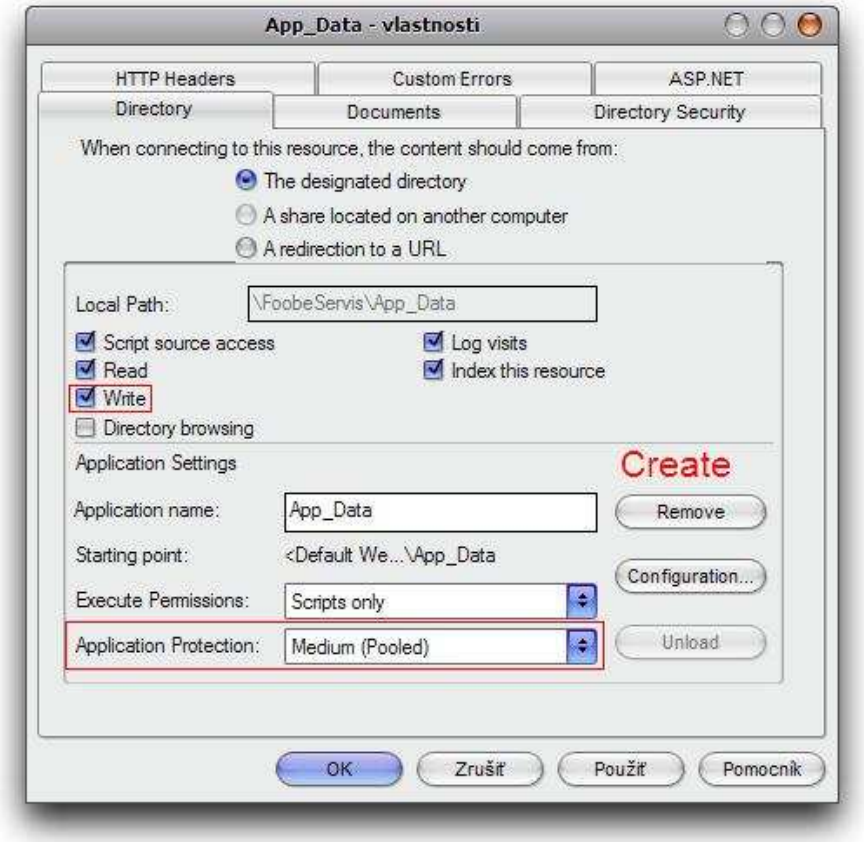
4. Krok 3.3 opakujte pre hlavný adresár služby *FoobeServis*.

5. Nastavte databázu MS SQL Server 2005 nasledovne:

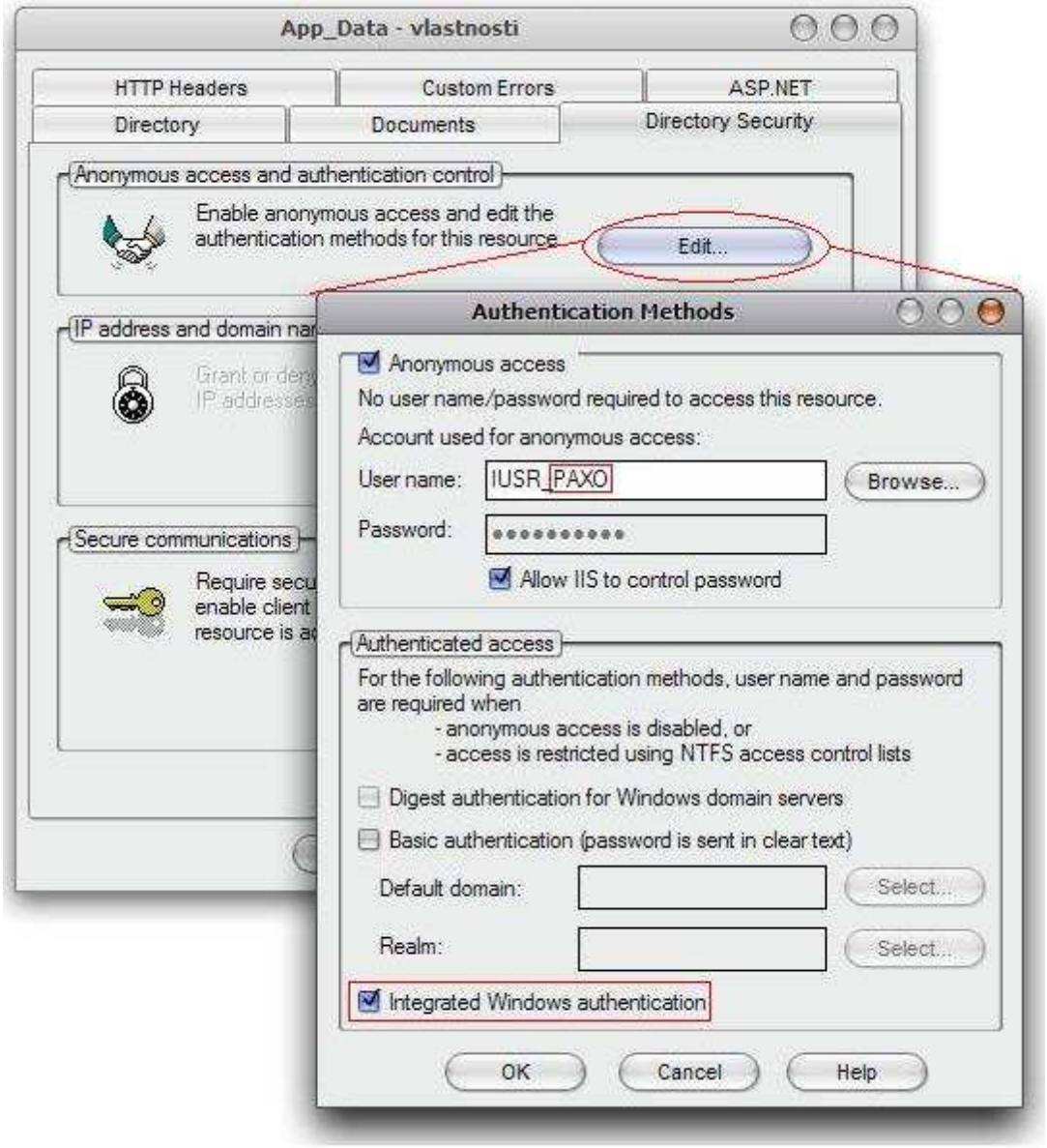
   5.1. Pripojte databázu zo súboru *foobeDB.mdf* na existujúci databázový server cez *SQL Server Management Studio Express*. Návod je na obr. 4.
   5.2. Zmeňte nastavenia pripojenia databázy v súbore *web.config* umiestnenom v adresári služby podľa obr. 5.



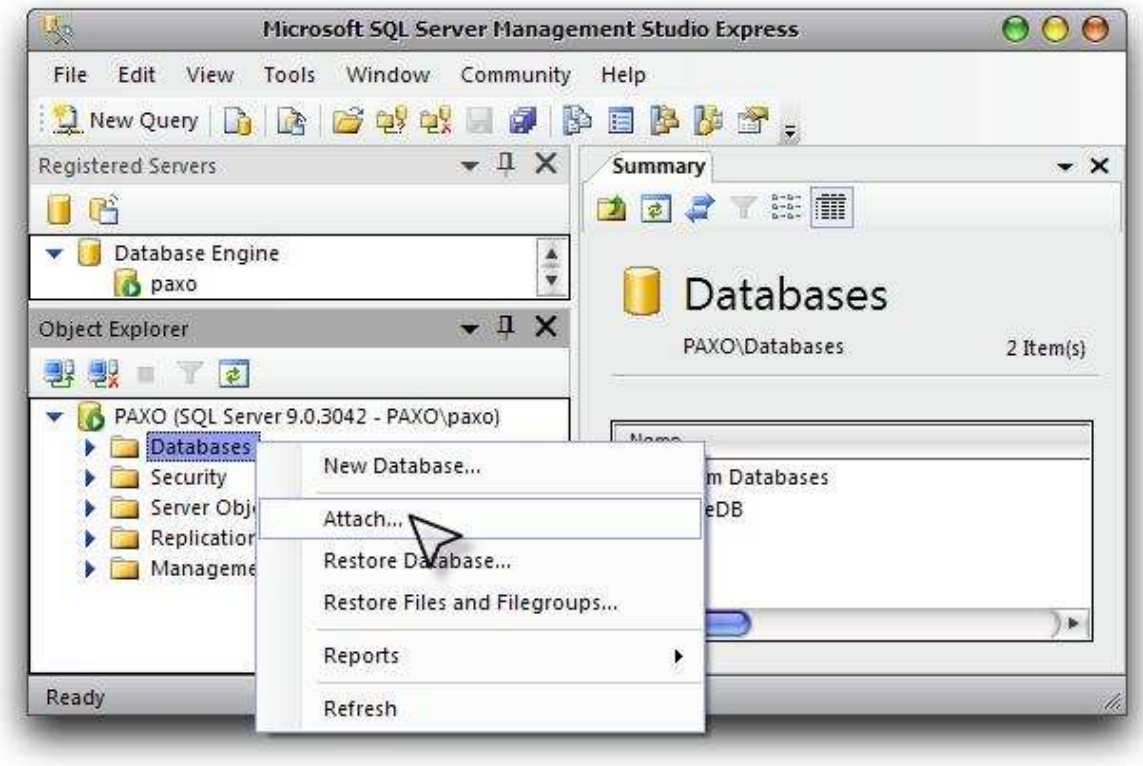
Obr. 1. Otvorenie nastavení adresára *App_Data*



Obr. 2. Nastavenie vlastností adresára *App_Data*



Obr. 3. Nastavenie používateľa pri prístupe do adresára *App_Data*



Obr. 4. Pripojenie databázy *foobeDB* na SQL Server

```
<?xml version="1.0"?>
<configuration>
  <appSettings>
    <add key="logFoobe" value="logFoobe.txt" />
    <add key="logServis" value="logServis.txt" />
    <add key="adresarIndex" value="index" />
    <add key="dbServer" value="paxo" />
    <add key="dbNazov" value="foobeDB" />
    <add key="dbLogin" value="ABCDEF" />
    <add key="dbHeslo" value="abcdef" />
  </appSettings>
  ⋮
```

Obr. 5. Nastavenie pripojenia databázy pre službu *FoobeServis* v súbore *web.config*

B.2 Inštalácia klientských aplikácií

Systém poskytuje dve aplikácie komunikujúce so službou *FoobeServis*. Prvá – administratívna - je určená na spravovanie používateľov, pridávanie a odstraňovanie dokumentov z bázy znalostí, a druhá umožňuje vyhľadávať v tejto báze.

Inštalácia administratívnej časti FoobeGui

1. Rozbaľte súbor *FoobeGuiAdmin.zip* do ľubovoľného adresára
2. Pre spustenie aplikácie otvorte súbor *FoobeGuiAdmin.exe*

Inštalácia vyhľadávacej časti FoobeGui

3. Rozbaľte súbor *FoobeGui.zip* do ľubovoľného adresára
4. Pre spustenie aplikácie otvorte súbor *FoobeGui.exe*

Príloha C – Revízia dokumentu

V tejto kapitole sa nachádzajú modifikácie a doplnenia špecifikácie požiadaviek. Druhá časť obsahuje fyzický model údajov.

C.1 Sumarizácia modifikácií a doplnení kapitoly 1. Analýza

Strana	Kapitola	Predmet zmeny	Popis zmeny
4	1.2	Slovo „vrstvovej“	Nahradenie slovom „viacvrstvovej“
8	1.3	Slovo „užívateľovej“	Nahradenie slovom „používateľovej“
8	1.3	Slovo „uživateľovu“	Nahradenie slovom „používateľovu“
8	1.3	Slovo „užívateľa“	Nahradenie slovom „používateľa“
8	1.3	Slovo „užívateľovi“	Nahradenie slovom „používateľovi“
8	1.3	Slovo „užívateľskú“	Nahradenie slovom „používateľskú“
15	1.4	Popis tabuľky 2	Nahradenie popisom „Požiadavky na vyhľadávanie v dotLucene“
22	1.7	Veta „Výhodou využitia takýchto prepojení je fakt že sa nám môže podariť nájsť dokument ktorý síce priamo nesúvisí s nami vyhľadávaným výrazom, ale odkazujú sa naň dokumenty ktoré s našou požiadavkou priamo súvisia a teda je veľká pravdepodobnosť, že takto nájdený dokument bude relevantný“	Nahradenie vetou: „Výhodou využitia takýchto prepojení je fakt, že sa nám môže podariť nájsť dokument, ktorý síce priamo nesúvisí s nami vyhľadávaným výrazom, ale odkazujú sa naň dokumenty, ktoré s našou požiadavkou priamo súvisia, a teda je veľká pravdepodobnosť, že takto nájdený dokument bude relevantný.“
23	1.7	Časť textu: „priradená hodnota určujúca jeho dôležitosť a tak ovplyvniť výsledky určenie relevancie dokumentu.“	Opravenie textu: „priradená hodnota určujúca jeho dôležitosť, a tak ovplyvniť určenie relevancie dokumentu.“

C.2 Sumarizácia modifikácií a doplnení kapitoly 3. Návrh

Strana	Kapitola	Predmet zmeny	Popis zmeny
30	2.4	Obr.11 Logický model údajov v systéme	Nahradenie diagramu novým
31	3.1	Obr.13 Architektúra systému	Nahradenie diagramu novým
36	3.3	Text „identifikátor dokumentu v databáze!“	Odstránenie výkričníka na konci vety
38	3.3	Veta „Metóda vracia množinu tzv. „trafení“ (trafenie obsahuje identifikátor dokumentu, relevanciu, a pod.)...“	Nahradenie danej vety: „Metóda vracia množinu nájdených dokumentov (jedna položka obsahuje identifikátor dokumentu, relevanciu, a pod.)“